Πρόγραμμα Μεταπτυχιακών Σπουδών

στην Αναλυτική των Επιχειρήσεων και Επιστήμη των Δεδομένων

Τμήμα Οργάνωσης και Διοίκησης Επιχειρήσεων

Διπλωματική Εργασία

Traffic Forecasting

του

ΚΟΠΡΟΥΤΣΙΔΗ ΠΑΥΛΟΥ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του μεταπτυχιακού διπλώματος στην Αναλυτική των Επιχειρήσεων και Επιστήμη των Δεδομένων

Αύγουστος 2022

Acknowledgements,

I would like to express my deepest grattitude to Professor Konstantinos Tarabanis and Assistant Professor Evangelos Kalampokis for their exceptional mentorship and unwavering support throughout my thesis project. Their vast knowledge and expertise have been invaluable to me and have greatly contributed to the successful completion of my thesis. I am truly grateful for their guidance and assistance, which have been instrumental in helping me overcome various challenges. Their significant contributions to my work have been a source of inspiration and motivation to me.

Furthermore, I would like to thank Mr Petros Brimos and Ms Areti Karamanou for their assistance in various stages of my research. Their help and expertise have been extremely important in completing my thesis.

Abbreviations

ADS: Atmosphere Data Store

AI: Artificial Intelligence

ANN: Shallow Artificial Neural Network

API: Application programming interface

AR: auto regressive

ARIMA: Autoregressive Integrated Moving Average

CDS: Climate Data Store

ITS: Intelligence transformation Systems

KNN: K-Nearest Neighbours

LIME: Local Interpretable Model-agnostic Explanations

MAE: Mean Absolute Error

MA: moving average

MAPE: Mean Absolute Percentage Error

ML: Machine Learning

NETCDF: Network Common Data Form

RMSE: Root Mean squared error

SHAP: Shapley Additive Explanations

SVM: Support Vector Machine

XAI: Explainable Artificial Intelligence

Abstract

Traffic congestion is a growing problem in many urban areas across the world, and Greece is no exception. In fact, traffic congestion in Greece's largest cities, Athens and Thessaloniki, is a significant issue that affects millions of commuters every day. A combination of factors such as high population density, limited transportation infrastructure, and a significant number of vehicles on the road all contribute to traffic congestion in these areas. Therefore, accurate prediction of the number of cars on a road network is crucial for effective traffic management. In recent years, machine learning has emerged as a valuable tool for traffic forecasting, providing precise forecasts for traffic conditions on a given day. However, there is often a lack of trust in machine learning outcomes, which can limit their usefulness in real-world applications. This is where Explainable Artificial Intelligence (XAI) comes in. XAI techniques can be used to clarify the reasoning behind machine learning models, making their outcomes more understandable and trustworthy. In this paper, we aim to forecast and explain the number of cars on a specific road in Athens for the next two hours, using machine learning and XAI techniques. We present a case study using hourly observations of the number of cars and their average speed, sourced from open data platforms like http://data.gov.gr/ and weather data from Copernicus, the European Union's Earth observation program. The XGBoost algorithm is used to create the forecasting model, and the SHapley Additive exPlanations (SHAP) framework is used to explain it. Our model has an MAE value of 325.12, indicating higher accuracy than the baseline model with an MAE of 542.4. The most significant factors affecting the number of cars is the previous two hours mean, the cos_seg_hour, the previous_two_hours_max, and the average speed. The SHAP values show that these variables have a significant impact on the number of cars on the road. The importance of traffic forecasting cannot be overstated, as it plays a critical role in various applications such as traffic management systems, intelligent transportation systems, and emergency response planning. Effective forecasting can help reduce travel times, improve safety, and optimize transportation infrastructure usage, ultimately improving the quality of life for citizens and businesses.

Table of Contents

List of Figures

List of Tables

1. Introduction

In recent times, there has been notable growth in the traffic and transportations sector considering the number of vehicles on roads. According to the Federal Highway Administration, there are over 276 million vehicles using the highways as of 2019 [24]. This ever-increasing load of road traffic has caused many serious transportation and health problems, rendering the employment of effective traffic control measurements necessary. Traffic forecasting, the process of estimating the future traffic state of a network given its past and future state, is a promising approach in terms of improving the usage and control of the existing road network. Such forecasts can be important for many applications in the traffic and transportations sector, for example reliable and highly accurate predictions of traffic quantities can contribute to the mitigation of congestion and to the reduction of vehicle pollutants that can both harm public health and induce climate change due to greenhouse gas emissions. Additionally, government agencies engaged in tourism can utilise accurate traffic predictions in order to improve their route guidance or alleviate travel congestion. Intelligent Transformation Systems (ITS) offers effective and accurate traffic predictions regarding all elements of a traffic network several steps ahead in time. Given its high importance, traffic forecasting has been extensively studied and a variety of models have been suggested in the relevant literature. These models can be either statistically based or machine learning based. A very commonly used statistically based technique that has been used for traffic forecasting is the autoregressive integrated moving average (ARIMA) model. Even though theoretically sound, the model fails to attain satisfactory results, as it cannot handle the nonlinear nature of traffic conditions datasets (i.e., peak periods). Considering other machine learning algorithms such as support vector machines (SVM) and k-Nearest neighbours, the algorithms perform considerably better but they still face difficulties in generating accurate results when they deal with enormous amounts of data. During the last few years, deep learning approaches are steadily adopted for traffic volume prediction. Despite their accuracy and reliability such approaches suffer in interpretability and are computationally expensive, restricting their deployment and usefulness in traffic control. In order to find a good balance between performance, interpretability and computational efficiency, extreme gradient boosting (XGboost) is a good approach for traffic volume prediction. XGboost has been employed and shown to be effective for time series prediction problems. For instance, XGboost has been used for stock price forecasting [26], as well

as daily electricity consumption prediction, since it can capture the nonlinear relationship between attributes and achieve better performance when it comes to nonlinear traffic data [27].

2. Background

2.1. Machine Learning

Machine learning (ML) is a fascinating and rapidly evolving field of Artificial Intelligence (AI) that is transforming the way we interact with technology. Through the use of complex algorithms and statistical models, ML enables computer systems to learn and improve from experience without the need for explicit programming. Unlike traditional computer programming, where a programmer writes code that explicitly defines how a computer system should perform a task, in machine learning, the computer system learns how to perform the task by analyzing and learning from data. The process of machine learning usually begins with the collection of relevant data for the task at hand, which can range from images and text to numerical data. Once the data has been collected, it must be preprocessed and cleaned to remove any noise or errors and converted into a suitable format for the machine learning model. The next step is to select or extract features from the data that are relevant to the task at hand and can be used to train the model. This helps to improve the accuracy of the model and ensure that it can predict the desired output. After the above steps, the model is trained using a machine learning algorithm, which helps it to learn from the data and create a model that can predict the desired output. Model evaluation is an essential step in the machine learning process, where the model is tested on a set of data that was not used for training. This process helps to measure the accuracy and performance of the model and ensures that it is ready to be deployed into a system to make predictions or decisions based on new data. Machine learning has a wide range of applications, including image and speech recognition, natural language processing, fraud detection, and predictive maintenance. It has the potential to revolutionize many industries and improve our lives by making systems more efficient, accurate, and intelligent. As technology continues to advance, machine learning will undoubtedly become an increasingly important field, driving innovation and improving our understanding of the world around us.

2.2. Supervised Machine Learning

In supervised learning, the model is trained using data that has been accurately labelled. The machine learning algorithm operates by giving a portion of the data to be trained on. The training dataset includes inputs and correct outputs, which enables the model to learn over time. Finally, after the training stage is completed the remaining portion of the dataset is used to test our trained model with unseen data and evaluate its performance. Supervised learning is separated into two categories of problems, Classification and Regression.

Classification uses an algorithm to accurately assign test data into specific categories. It identifies specific entities within the dataset and attempts to draw some conclusion on the way that those entities should be labelled or defined. The most common classification algorithms are linear classifiers, support vector machine (SVM), decision trees, k-nearest neighbours and random forest [61] [62].

Regression is used to understand the relationship between dependent and independent variables. It is commonly used to make projections, such as for sales revenue for a given business. Linear regression, logistic regression, and polynomial regression are popular regression algorithms.

2.3. Unsupervised Machine Learning

In contrast to supervised learning, unsupervised learning can use unlabelled data as input and output. This means that computer comprehension of the data does not require human interaction. Unsupervised learning cannot process labels since it discovers patterns and differences in data by building hidden structures [63]. Clustering is one of the most common types of unsupervised learning. Clustering algorithms group similar data points together based on their proximity in feature space. Other unsupervised learning techniques include principal component analysis (PCA), which is used for dimensionality reduction, and association rule mining, which is used to discover relationships between variables in a dataset. Some common applications of unsupervised learning include anomaly detection, where the algorithm is used to identify unusual or abnormal data points, and market segmentation, where the algorithm is used to group customers based on their buying behaviour. Unsupervised learning can be a powerful tool for data analysis and exploration, but it can also be challenging because there is no clear objective or goal

to optimize. The success of an unsupervised learning algorithm often depends on the quality of the data and the ability of the algorithm to identify meaningful patterns and structures.

2.4. Decision Trees

Decision trees is an algorithm commonly used in machine learning, for both classification and regression problems, they are easy to use, and they are often a good exploratory method if you are interested in getting a better idea about what the influential features are in your dataset. The goal when building a decision tree is to find the sequence of questions that has the best accuracy at classifying the data in the fewest steps. A decision tree contains decision nodes which contain a condition to test the value of an attribute, edges which depending on the outcome of the test, connect with the next node, and leaf nodes which predicts the outcome, all the above combined consists a complete structure of decision tree as shown below fig. 1 [56]. Once a decision tree has been constructed, it can be used to make predictions by traversing the tree from the root to a leaf node based on the values of the input variables. The decision or prediction associated with the leaf node is then returned as the output. However, decision trees can be prone to overfitting if the tree is too deep or if the training data is noisy or unbalanced. To overcome this problem, various techniques such as pruning, ensemble methods, and random forests can be used

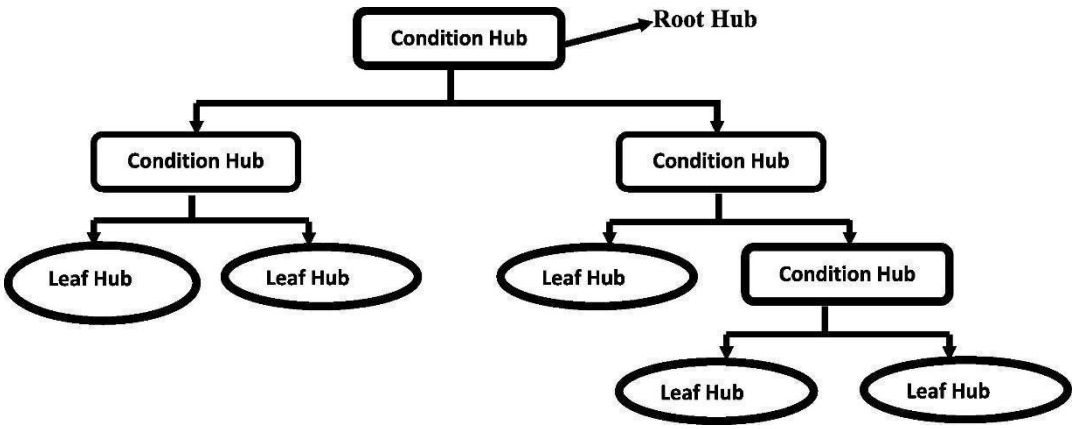


Figure 1 Decision Tree Structure

2.5. Random Forests

Random decision forests are an ensemble method that can be used for both regression and classification problems, they operate by constructing a number of decision trees at training phase and give as an output the class that is the mode of the classes (classification) or the mean (regression) of the individual trees. Random decision forests reduce the risk of overfitting to the training set that decision trees tend to deal with [55]. Each decision tree in a random forest is trained on a bootstrap sample of the training data, meaning that some of the data may be repeated and some may be left out. Additionally, at each node of the tree, a random subset of the input features is selected to split the data, rather than using all features. This helps to reduce overfitting and improve generalization. Once the individual decision trees have been trained, the random forest combines their predictions using a majority vote for classification or averaging for regression. The resulting prediction is typically more accurate and less prone to overfitting than the prediction of a single decision tree. Random forests are a popular choice for classification and regression tasks in machine learning because they are easy to use, highly scalable, and robust to noise and outliers the data. They can also provide information about the importance of each input feature in making predictions, which can be useful for feature selection and interpretation. Some common applications of random forests include image classification, text classification, and predicting customer churn in business. However, random forests can be computationally expensive and may require tuning of hyperparameters to achieve optimal performance.

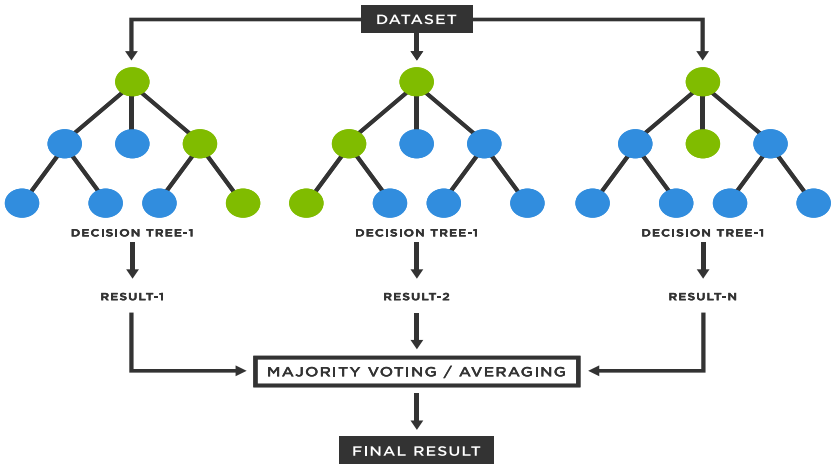


Figure 2 Random Forests Structure

2.6. Extreme Gradient Boosting (XGboost)

The XGboost algorithm, is based on gradient decision trees and utilises a second-order Taylor expansion in order to calculate the loss function and it can perform excellently regarding both computational speed and model accuracy. It is an ensemble method that combines the outputs from individual decision trees, however gradient boosted trees and random forests differ in the way the individual trees are built and in the way the results are combined. Random forests build independent decision trees and combine them in parallel, on the other hand, gradient boosted trees use a method called boosting. Boosting combines weak learners sequentially, so that each new tree corrects the errors of the previous one, weak learners are usually decision trees with only one split called decision stumps. The evaluation of each tree's performance is done by using a loss function [57]. XGBoost uses a customized loss function to optimize the performance of the model during training. The loss function measures the difference between the predicted values and the true values and guides the optimization algorithm to adjust the model parameters in a way that minimizes this difference. The default loss function for classification problems in XGBoost is the softmax function, which is used to compute the probabilities of each class. For regression problems, the default loss function is the mean squared error (MSE). In addition to the default loss functions, XGBoost provides several other loss functions that can be used depending on the specific problem and the desired performance metrics. Some of the popular loss functions used in XGBoost include, binary logistic loss which is used for binary classification problems, where the goal is to predict the probability of a sample belonging to one of the two classes. Multi-class logistic loss, for multi-class classification problems, where the goal is to predict the probability of a sample belonging to one of the multiple classes. Poisson regression loss, for regression problems where the target variable is counting data and follows a Poisson distribution. Gamma also for regression problems where the target variable is continuous and positive and finally Huber loss which is a robust loss function that is less sensitive to outliers compared to the mean squared error. The choice of the loss function depends on the specific problem and the desired performance metrics. XGBoost provides flexibility in choosing the appropriate loss function to optimize the performance of the model. Considering binary classification problems cross entropy loss is the dominant loss function, mathematically is defined as follows:

$L = -\sum_{i=1}^{2} t_i \log(p_i)$

Where t_i is the true value taking a value 0 or 1 and p_i is the softmax probability for the i^{th} class.

As for the regression problems the default loss function is mean absolute error (MAE) and it is defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i|$$

Where y_i is the prediction value, x_i is the true value and n is the total number of data points.

2.7. Time Series Forecasting in Machine Learning

Time series forecasting is a type of machine learning problem that involves predicting future values of a time series, based on historical data. A time series is a sequence of data points measured over time, such as stock prices, temperature readings, or website traffic. The goal of time series forecasting is to build a model that can accurately predict future values of the time series, given its past values. This can be used for a variety of applications, such as predicting future sales, forecasting weather patterns, or predicting website traffic. Time series forecasting is a challenging problem, as it requires handling the unique properties of time series data, such as trend, seasonality, and autocorrelation. However, it can be a valuable tool for predicting future values of a time series and making informed decisions based on that information. The main difference between a time series split and a normal data split is the way the data is split and the way the splits are evaluated. In a normal data split, the data is randomly divided into training and testing sets, and the model is trained on the training set and evaluated on the testing set. The goal is to ensure that the model can generalize well to new data, i.e., data that was not used during training. In a time, series split, on the other hand, the data is split in a way that preserves the order of the time series. This is important because the goal of time series forecasting is to predict future values based on past values. Therefore, the training set contains only past data, while the testing set contains future data. This ensures that the model is evaluated on data that it has not seen during training, but that is still consistent with the problem it is trying to solve. Additionally, in time series forecasting, there is often a need to consider the temporal dependence between the data points, such as autocorrelation and seasonality. This can affect the choice of model and the evaluation metrics used to measure model

performance. For example, in time series forecasting, it is common to use metrics such as mean absolute percentage error (MAPE) or mean squared error (MSE) that take into account the magnitude and direction of errors relative to the true values.

2.8. Explainable AI

One of the key benefits of XAI is that it can help to mitigate the potential risks associated with the use of AI. For example, XAI can help to identify and prevent bias in AI systems by providing explanations for how the system made its decisions. In addition, XAI can help to improve the interpretability and explainability of AI systems, which is particularly important for domains such as healthcare and finance where the consequences of an incorrect decision can be significant. Another important aspect of XAI is that it can improve collaboration between humans and machines. By providing clear and transparent explanations of their decisions, AI systems can work together more effectively with human users. This can lead to more accurate and efficient decision-making, as well as improved user satisfaction and trust in the system.as XAI continues to evolve, it is likely that new techniques and approaches will be developed to enhance the interpretability and transparency of AI systems. For example, research in natural language processing and visualization techniques may enable AI systems to provide even more intuitive and accessible explanations. Ultimately, the goal of XAI is to build AI systems that are not only accurate and reliable but also transparent, trustworthy, and ethical.

2.8.1. Shapley Additive Explanations (SHAP)

Lloyd Shapley first suggested the technique for getting Shapley values in 1953 (‘A value for n-Person Games L.S. Shapley. 1953’). Shapley values are contributions that are made in a situation where ‘n’ players work together to achieve a prize ‘p’ that is meant to be equally divided to each of the ‘n’ players according to their individual contributions. To put it simply, a Shapley value is the average marginal contribution of a feature instance across all potential coalitions. Shapley Additive Explanations (SHAP) is a method introduced by Lundberg and Lee (‘A unified approach to interpreting model predictions. 2017’) for the understanding of machine learning models through Shapley values. SHAP (SHapley Additive exPlanations) values are a method for explaining the output of machine learning models. They provide a way to understand the contribution of each feature to the model's output for a specific prediction. SHAP values can be used to understand how a model makes predictions and identify areas where the model may be

making incorrect or biased predictions. They can also be used to identify features that are important to the model and help identify any potential issues with the data or model. To calculate SHAP values, the model's output is decomposed into a sum of the contributions of each feature, with each feature's contribution being based on its importance and the prediction being made. The contributions are based on the Shapley values from game theory, which represent the average marginal contribution of a player to the overall outcome of a game. SHAP values have several advantages over other methods for interpreting machine learning models. They are model agnostic, meaning they can be used with any machine learning model, and they provide a global explanation of the model's output, rather than just a local explanation. They also have a strong theoretical foundation and have been shown to be really accurate and robust.

2.8.2. Local Interpretable Model-agnostic Explanations (Lime)

LIME (Local Interpretable Model-agnostic Explanations) is a method for explaining the predictions of any machine learning model. It is model-agnostic, which means that it can be used to explain the predictions of any type of machine learning model, regardless of the underlying algorithms or techniques used. The basic idea behind LIME is to approximate the behavior of the model in a small region around a specific prediction and explain that behavior in terms of simple, interpretable features. LIME does this by fitting a simple, interpretable model (such as a linear regression or a decision tree) to the model's predictions in the vicinity of the target prediction. This simple model is then used to explain the prediction, by showing how each feature contributes to the prediction. LIME is designed to provide local explanations, which means that it provides an explanation for a single prediction, rather than an explanation of the entire model. This allows LIME to generate explanations that are easy to understand and interpret, even for complex models. LIME is implemented in several programming languages, including Python, and is available as an open-source library. It is widely used in a variety of applications, including natural language processing, computer vision, and image classification. It is also useful for helping to understand the decisions made by black-box models, such as deep neural networks, that can be difficult to interpret.

2.8.3. Feature Importance

Feature importance is a technique used in Artificial Intelligence (AI) to understand the relative importance of input features or variables in a model. It helps to identify which features or variables have the most significant impact on the model's output or predictions. In machine learning models, the algorithm learns patterns and relationships between input features and output variables from training data. Some features may be more relevant than others in predicting the target variable, and feature importance measures can help to identify these important features. There are various techniques to calculate feature importance, depending on the type of model used. For example, in decision tree models, feature importance is calculated by measuring the decrease in impurity or entropy in the model when a particular feature is used for splitting the data. In linear models, feature importance can be measured by the magnitude of the coefficients assigned to each feature. The interpretation of feature importance can help to gain insights into the model's decision-making process and improve the understanding of the relationship between input features and output variables. It can also be useful for feature selection, where only the most relevant features are used in the model, reducing the model's complexity and improving its performance. In summary, feature importance is an important concept in AI and machine learning, enabling us to understand the importance of input features and improve the interpretability and performance of AI models.

2.8.4. Traffic explainable AI

Traffic explainable AI, also known as XAI in the field of traffic management and transportation systems, is a rapidly developing area of research that focuses on using explainable AI techniques to provide insights and explanations about the decisions made by AI systems in traffic-related tasks. While AI systems are widely used in transportation systems to make decisions, such as predicting traffic congestion, optimizing traffic flow, and managing traffic signals, it is essential to understand how the AI system arrived at a particular decision and why it made that decision. Traffic explainable AI provides insights into the decision-making process of AI systems in traffic management and transportation systems, enabling stakeholders to trust and validate the AI system's decisions. The use of XAI techniques in traffic management and transportation systems improves transparency, accountability, and trust in AI systems. By understanding the decision-making process of AI systems, stakeholders can identify and correct any biases or errors in the system's

decision-making process. One of the most significant benefits of traffic explainable AI is the ability to identify the most important variables in the AI system's decision-making process. Feature importance techniques can be used to identify which variables had the most significant impact on the model's output. Furthermore, SHAP additive explanations can show how changes to variables affect the AI system's decision. Overall, traffic explainable AI is a vital area of research in transportation systems. It can help to improve the safety, efficiency, and reliability of traffic management and transportation systems by providing stakeholders with insights and explanations about the decisions made by AI systems. This increased understanding of the decision-making process can lead to better outcomes and help to build trust in AI systems.

3. Related Works

A variety of traffic forecasting models have been proposed in the relevant literature. They can be classified into three categories, the parametric, the non-parametric and the hybrid models.

3.1. Parametric Models

Parametric models were the first to be used for traffic forecasting. They are statistical time series models, characterised from their predefined structure and the fact that only the values of their parameters have to be estimated from the available traffic data. Most of these parametric models are based on the **AR (auto regressive)** model; this type of model tries to predict the variable of interest by using a linear combination of its past values. The term autoregression shows that it is a regression of the variable against itself. Mathematically, it is defined as follows:

$$y_t = c + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \ldots + \beta_p y_{t-p} + \varepsilon_t$$

Where c is a constant value, β represents the AR parameters (magnitude of the correlation), p represents the number of lags and ε_t is the error [28]

The **MA (moving average)** model is a common approach for modelling time series. It basically expresses that the present value is a linear combination of the mean of the series, the present error term and past error terms. The formulation of the MA model is defined as follows:

$$y_{t=} c + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q},$$

Where y_t, represents a value forecast by lagged values of the residual error (ε_t), θ represents the MA parameter value (autocorrelation of errors), and q is the number of lags.

The Autoregressive Moving Average Models **ARMA** is a combination of AR and MR models and is defined as follows:

$$y_t = C + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \ldots - \theta_q e_{t-q} + e_t$$

AR, MA and ARMA models require the time series to be stationary. This means that the time series needs to have a constant mean, constant variance, and no seasonality over time. In order to overcome this assumption, we use integration.

Integration (I) represents the difference of raw observations to allow the time series to become stationary, this can be achieved by replacing the original data values with the difference between them and their previous values. For instance, first-order differencing has a linear trend and is given as $Y`_t = y_t - y_{t-1}$, second order differencing has a quadratic trend and is given as $Y``_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$. For generality purposes it is developed using the L lag-$_{operator}$, which is defined as $Ly_t - y_{y-1}$ so first and second order differencing are represented as $(1 - L)y_t$ and $(1 - L)^2 y_t$, respectively. A d^{th}-order differencing order can be given as:

$$y^d_t = (1 - L)^d y_t$$

Where d, represents the degree of difference that is necessary for stationarity (The number of times the data are differenced).

Autoregressive Integrated Moving Average (ARIMA) is a statistical model to forecast and analyse time series data, it was introduced by Box and Jenkins (1976). ARIMA is developed based on AR and MA models as well as Integration. The formulation of the ARIMA model is defined as follows:

$$y_t = c + \varphi_1 y_{t-1}^d + \varphi_2 y_{t-2}^d + \ldots + \varphi_p y_{t-p}^d + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \ldots - \theta_q e_{t-q} + e_t$$

The basic parameters of the ARIMA model are p, d, and q. Where p is the number of autoregressive terms, d is the number of nonseasonal differences needed in order to achieve stationarity, and q is the number of lagged forecast errors in the prediction equation [28].

Considering the parametric models, Alghamadi, Elgazzar, Bayoumi, Sharaf and Shah (2019) [33] proposed an autoregressive integrated moving average (ARIMA) approach on traffic forecasting on non-Gaussian traffic data. Diamantopoulos, Kehagias, Konig and Tzorvas (2013) [34] developed an ARIMA-based spatiotemporal traffic forecasting model that makes use of the Pearson correlation coefficient in order to predict the correlations between time series. Considering graph theory on correlation estimation in large-traffic scale networks, Salamanis, Kehagias, Filelis-Papadopoulos, Tzovaras, and Gravvanis (2016) introduced a technique based on spatial graphs [35]. In the same direction Duan, Mao, Liang and Zhang (2018) [36] found the appropriate lag between the traffic time series before moving to the estimation of correlations. Furthermore, Guo, Huang, and Williams

(2014) [37], used Kalman filters to create a combination of ARIMA (SARIMA) and generalised autoregressive conditional heteroskedasticity (GARCH) model for short-term traffic forecasting. From another point of view, Abadi, Rajabioun, and Ioannou (2015) [38] implemented an autoregressive model for short-term traffic forecasting when the available data are limited. In addition, Kamarianakis, Shen, and Wynter (2012) [39], implemented the least absolute shrinkage and selection operator (LASSO) method for both model selection and regularisation. The majority of all these parametric forecasting models are based on the classic Autoregressive model (AR).

3.2. Non-Parametric Models

The non-parametric models do not have a predefined structure thus both their structure and their parameters have to be estimated from the data. The majority of non-parametric models are based on traditional machine learning methods, such as k-nearest neighbours (KNN), support vector regression (SVR) and shallow artificial neural networks (ANN) [22].

K-Nearest Neighbours (KNN) is a non-parametric, supervised machine learning algorithm that can be used for both classification and regression predictive problems. The main KNN algorithm assumption is that similar things exist in close proximity. KNN captures this idea of similarity by calculating the distance between points on a graph. There are many ways to calculate distance, however the most commonly used is the straight-line distance also known as Euclidean distance. The K refers to the number of nearest neighbours the classifier will retrieve and use in order to make its prediction [29]. The choice of the value of k can have a significant impact on the performance of the algorithm. A small value of k can result in overfitting, while a large value of k can result in underfitting. The optimal value of k depends on the nature of the data and the specific task at hand.

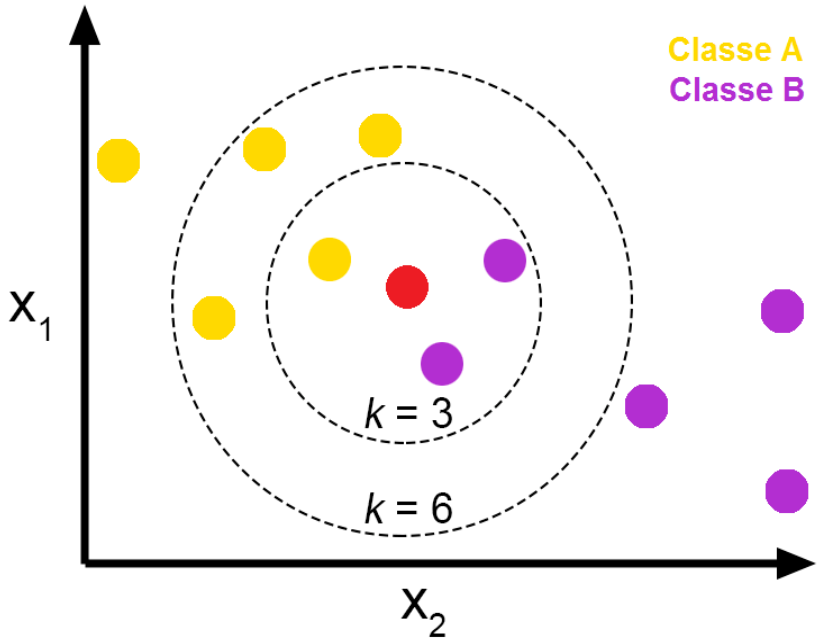


Figure 3 KNN Algorithm Classification of a new data point (red Colour) depending on the value ok K

Support Vector Machine (SVM), is a machine learning method based on statistical theory and structural risk minimization theory, proposed by Vapnik's research team for pattern classification and non-linear regression [30]. The application of support vector machines to non-linear problems is known as **support vector regression** (**SVR**), whose main objective is to fit the data points to a suitable hyperplane so that the error distance between the data points and the hyperplane is minimised, leading to a description and prediction of the non-linear relationship among the data points and the hyperplane [31]. SVM can handle both linearly separable and non-linearly separable data by using different kernel functions. A kernel function is a function that maps the input data into a higher-dimensional space where the data can be separated by a hyperplane. SVM is a powerful algorithm that has been widely used in various applications such as text classification, image classification, and bioinformatics. However, it can be computationally expensive when dealing with large datasets or complex kernel functions. In such cases, other algorithms such as neural networks may be more suitable.

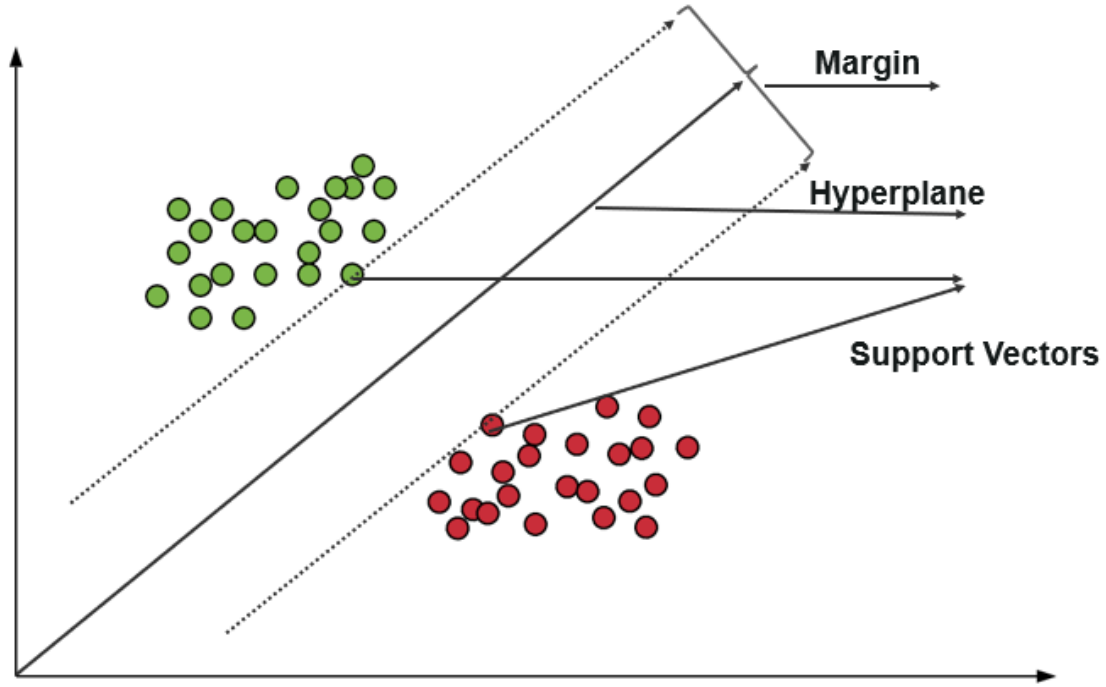


Figure 4 Support Vector Machine (SVM) Algorithm Illustration

Shallow Artificial Neural Network (ANN) is inspired by biological learning processes as it occurs in human brains. One of the main goals behind this is to build a highly intelligent system like a human brain [32]. Neural networks take in data, train themselves to recognize the patterns in this data and finally predict the outputs for a new set with similar data. Neural networks are made of layers of neurons, these neurons constitute the core processing units of the network, the first layer is called input layer and receives the input, the last layer is called the output layer and it's the one that predicts the final output. Between the input and the output layer are the hidden layers which perform most of the computations that are required in order to predict the final output. Each neuron is connected to another neuron through channels each of these channels is assigned a numerical value known as weight, the inputs are multiplied to the corresponding weights and the sum is sent as input to the neurons in the hidden layer, each of these neurons is associated with a numerical value called the bias, which is then added to the input's sum this value is then passed through a threshold function called the activation function. The result of this activation function determines if the particular neuron will get activated or not, an activated neuron transmits data to the neurons of the next layer through the channels. In the output layer the neuron with the highest value determines the output.

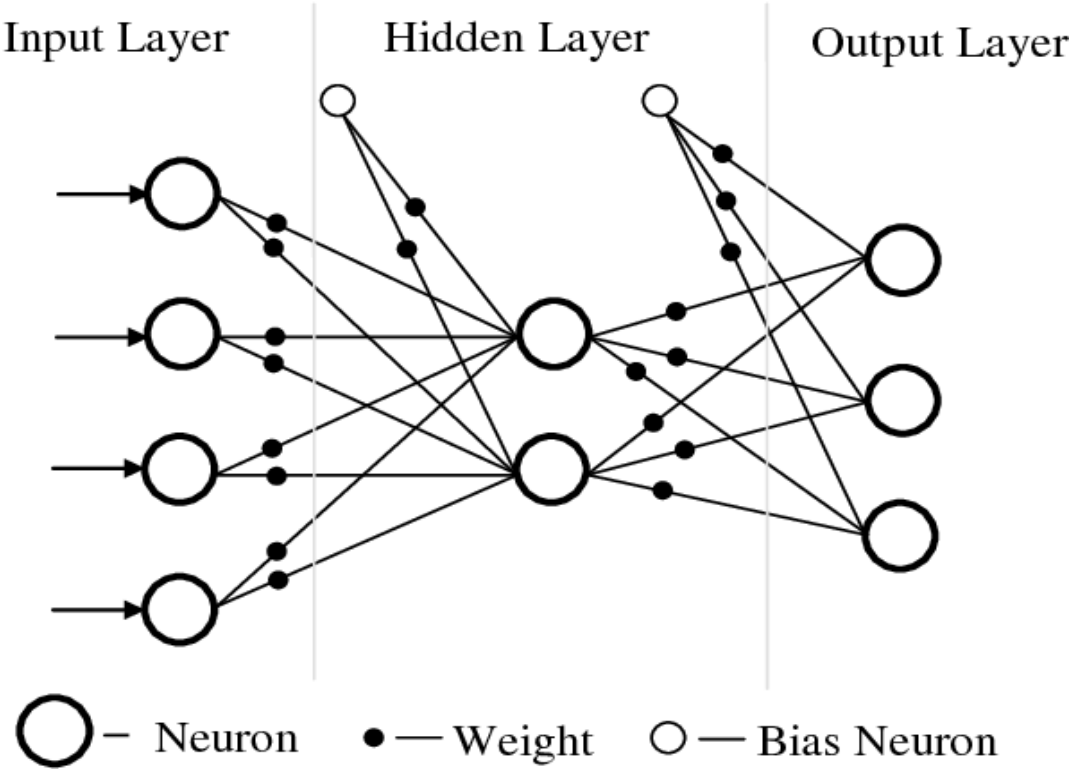


Figure 5 Basic ANN Architecture [58]

As for the non-parametric models, Zheng and Su (2014) [40], developed a kNN model for traffic forecasting that integrates novel methods to select the value of k nearest neighbours. Cai et al. (2016) [41], presented an improved kNN model that takes into consideration the spatiotemporal correlations among the roads of the traffic network to predict traffic. Furthermore, Sun, Cheng, Goswami, and Bai (2018) [42] proposed a dynamic self-adjustable kNN model for traffic prediction. On the other hand, Guo, Polak and Krishman (2018) [43], selected SVR as a machine learning model to predict short-term traffic by using a fusion-based forecasting. Hu, Yan, Liu, and Wang (2016) [44], proposed an SVR model that uses particle swarm optimization (PSO) to optimize the hyperparameters of the model, while Cong, Wang, and Li (2016) [45], used least square SVR for the prediction and the fruit fly optimization algorithm (FOA) in order to optimize the hyperparameters. Additionally, Yao et al (2016) [46], introduced an SVR model for short-term traffic forecasting that uses both the spatial and temporal characteristics of the traffic network. Moreover, Zhu, Cao, and Zhu (2014) [47], proposed a model that is based on a radial basis function network (RBFN) that integrates the traffic information of

adjacent intersections, in order to predict traffic at the intersection of interest. In the same context, Wang, Tsapakis, and Zhong (2016) [48] utilized a space-time delay neural network (SDTNN) that integrates spatiotemporal correlations between different elements of the traffic network. Furthermore, Fusco, Colombaroni, and Isaenko (2016) [49], used together two ANNs to accurately representing space and time correlations among traffic variables, whilst Tang, Liu, Zou, Zhang, and Wang (2017) [50] created and used an evolving fuzzy neural network (EFNN), for multi-step traffic forecasting.

3.3. Hybrid Models

Hybrid models for traffic forecasting combine different types of models to improve the accuracy of traffic predictions. These models often combine traditional statistical models, such as ARIMA or exponential smoothing, with machine learning techniques, such as neural networks or support vector regression. One example of a hybrid traffic forecasting model is the use of an Artificial Neural Network (ANN) to model the underlying relationships between the input data and the traffic output, combined with a time series model like ARIMA to capture the temporal dependence in the data. Another example is the combination of a machine learning model with a traditional statistical model, such as the combination of a Random Forest with an ARIMA model. The idea behind hybrid models is to take advantage of the strengths of different models and to overcome their weaknesses. For example, a machine learning model may be able to handle a large number of input variables, but may not be able to account for temporal dependencies in the data. A time series model, on the other hand, may be able to capture these dependencies, but may not be able to handle a large number of input variables. By combining these models, a hybrid model can improve the overall accuracy of traffic predictions. Considering the Hybrid models, Ladino, Kibanagou, de Wit, and Fourati (2017) [51] firstly clustered the available traffic time series into groups by using the k-means algorithm, and then formulated a computationally simple predictor for each group. Furthermore, Li, Zhai, and Xu (2017) [52], used a combination of an ARIMA model with a radial basis function network (RBFN), while Guo, Liu, Huang, Wei, and Gao (2018) [53], injected fuzzy traffic information into KNN and ANN models. Additionally, Raza and Zhong (2018) [54], combined an ANN and a locally weighted regression model (LWR) with genetic algorithms to achieve the best possible accuracy.

4. Weather Data from Copernicus AI System

Copernicus is the European Union's Earth observation program that inspects our planet and its environment to constantly be aware of its state and health. Copernicus information services are based on data from a constellation of 6 families of satellites, known as the Sentinels. These satellites are designed to meet the needs of the Copernicus system and its users. They can either operate alone or be combined with sensors placed on the seas, land or in the air [1]. Copernicus' main objective is to provide a large amount of reliable and up to date open-access data based on satellite information and in situ (on site) observations. The program is supervised and coordinated by the European Commission (EC) but it is applied through a series of agreements with international organisations. The European Centre for Medium Range Weather Forecasts (ECMWF) has been assigned by the European Union to implement the Copernicus Climate Change Service **C3S** and the Copernicus Atmosphere Monitoring Service (**CAMS**). The Copernicus services give value to all these satellites and in situ data by processing and analysing them. A variety of datasets from many years ago are made comparable and searchable. As a result, patterns can be examined and used to improve forecasts. All these value-adding activities are streamlined through six Copernicus services.

4.1. The Six Sentinels

Sentinel 1

The first of the Copernicus Sentinels is a constellation of two radar imagery satellites in the same orbit, offering supply of images of Earth's surface throughout the day and night. Sentinel 1A was launched on 3 April 2014 whereas Sentinel 1B was launched on 25 April 2016. Their main applications include monitoring of sea ice and icebergs, as well as land-use change, agriculture and deforestation. Furthermore, support to emergency management is provided considering for example floods or earthquakes [7].

Sentinel 2

Sentinel 2 is a constellation of two identical satellites in the same orbit. These two satellites provide images of land and coastal areas at high spatial resolution. Sentinel 2A was launched on 23 June 2015 and Sentinel 2B on 7 March 2017. Their main usage includes agriculture, land ecosystems monitoring, inland and coastal water quality monitoring, disasters mapping and civil security [7].

Sentinel 3

The main objective of sentinel-3 mission is to measure sea surface topography, as well as sea and land temperature. The data provided is highly accurate and reliable and, consequently, support to ocean forecasting systems along with environmental and climate monitoring is provided. Sentinel 3A was launched on 16 February 2016 and Sentinel 3B on 25 April 2018 [7].

Sentinel 4

Sentinel-4 provides data considering atmospheric composition monitoring. Its main use is to control air quality trace gases and aerosols across all Europe at high spatial resolution [7].

Sentinel 5

Sentinel-5 Precursor's mission launched on 13 October 2017. Its mission is to provide data continuity until the launch of Sentinel-5 which is going to be dedicated to Copernicus atmospheric mission.

Sentinel 6

Sentinel-6 is also scheduled to be launched during the next few years. Sentinel-6 will provide high accuracy altimetry considering global sea surface height measurement, for operational oceanography and climate studies.

4.2. The Six Copernicus Services

C3S, contains a vast variety of climate observations, measurements, analysis tools and methods. The core of the C3S service is the Climate Data Store (CDS). CDS contains data about the past, present and future climate data freely available for users to explore.[2] The C3S mission is to support adaption and mitigation policies in order to make the impacts of climate change less severe, by offering a variety of consistent and reliable information. C3S is the first unified network of its kind, developed between agencies across all over the world, proving that a data-oriented approach can be used to reduce unpredictability and risk. C3S constitutes one of the first attempts to operationalize the generation of climate data by taking advantage of the experience of satellite remote sensing and the production of weather predictions.

CAMS, is the European Copernicus Atmospheric Monitoring Service that monitors the Atmospheric composition and offers analysis and forecasts to inform a wide variety of users from policymakers to citizens and businesses. CAMS products, mainly support planning and monitoring solar energy, gas emissions and air quality. As all Copernicus services, CAMS deliver measurements done in situ with the remote sensing imagery done by satellites [6].

CMEMS, is the Copernicus Marine Environment Monitoring Service, which offers systematic information on the physical and biogeochemical state of the ocean and marine ecosystems. This service contributes to the protection and the sustainable management of living marine resources, as well as, to monitoring water quality and control pollution [9].

CLMS, is the Copernicus Land monitoring Service that provides geographical information on land cover and its changes. It is applicable in a variety of sectors such as spatial and urban planning, forest and water management, as well as agriculture. CLMS is implemented by the European Environment Agency and the European Commission joint Research Centre (JRC) [10].

SECURITY, the Copernicus service regarding security aims to support European Union policies, by enhancing crisis prevention, preparedness and response in three different areas: Border Surveillance, Maritime Surveillance and EU external action supporting [11].

EMS, is the Copernicus Emergency Management Service that provides to everyone that is related to the management of natural disasters, man-made emergency situations and humanitarian crises, timely and accurate information produced by satellite remote sensing and completed by available on-site data sources.

4.3. CDS Toolbox

The Climate Data (CDS) Toolbox is a way to get access to a vast variety of past and feature climate information. Its objective is to link raw data to online computing power through a programming interface. Each user can use its own online workspace to create applications in Python and execute them on the CDS computers. This procedure gives the opportunity to the user to retrieve data, make calculations and display the results. In order to use the CDS toolbox, a basic working knowledge of Python is needed as well as having access to the internet. Finally, there is no need for a powerful computer or a lot of memory

space, as the calculations and the data processing take place online within the CDS environment.[8] The two available data formats that data can be retrieved are NETCDF and GRIB.

4.4. Data Formats

Network Common Data Form (**NETCDF**) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array shaped scientific data. In order to access and manipulate this kind of data into python, the netcdf4-python is used, which is a Python interface to the netCDF C library. This interface can be accessed by running the following command into a python script: '!pip install netcdf4' [4].

GRIB is a file format that is used for the storage and transport of gridded meteorological data. This file format was designed and it is also maintained by the World Meteorological Organization [12]. This type of data is mainly used by individuals and businesses that are interested in numerical data from weather models in the rawest possible form. By using xarray library and the cfgrib engine GRIB, the data can be easily visualised and analysed with Python.

4.5. Copernicus Usage

The data and the information provided by the six Copernicus services can be used by users for a wide range of applications in many different areas:

Agriculture, which is historically one of the first sectors to exploit Earth observation. Agriculture is probably considered the most promising market in terms of the impact of Copernicus. The wealth of information that Copernicus is offering helps assessing agricultural land use and trends, crop conditions and yield forecasts. It also supports input, farm, recording and irrigation management. [14]

In Austria, because of the shortage of precipitation, the area of Marchfeld is dealing with serious water management issues. Farmers have begun to increase crop production by irrigation using groundwater, and irrigation water accounts for about 60% of the total freshwater production in this area. From the 30 farmers that were interviewed, 50 % claimed that further improvements could be achieved if the total amount of water requirements and the distribution of individual irrigation events were optimized. 54% of the farmers expressed the willingness to pay for a remote sensing service. Stakeholders

made a comparison between the irrigation volumes estimated from satellite and the irrigation supplied by the farmers to estimate how efficiently the water is currently used. The service delivered crop development maps available every 7 to 10 days, with a spatial resolution between 10 and 20 metres. In addition, evapotranspiration maps, information and weather data as well as forecasts provided daily and finally specific irrigation requirements depending on crop types. The addition of Sentinel data in this specific irrigation management product will allow a cost reduction considering the images, which currently range from 15,000 EUR to 35,000 EUR. The service based on sentinel-2 data would have cost 1.25 EUR per hectare per year, comparing to commercial data that would range between 2.5 EUR to 4.3 per hectare year, which is a 70% reduction of the cost of a precision agriculture service [15].

Energy and Natural Resources, the data provided by Copernicus support applications such as solar and wind energy production forecasting, renewable energy site selection or water and biomass monitoring. As a result, Copernicus can support the efficient exploitation of renewable energies, which will contribute to meeting the global energy demands without the risk of increasing CO2 emissions.

Over the past two years the price of oil has been decreasing. This reduction of the oil price has negatively affected the penetration rate of earth observation (EO) products in the oil and gas (O&G) industry. The main problem that arises is the fact that O&G companies are not willing to invest in new products anymore with such a low price of oil. The earth observation for oil and gas (EO4OG) project consists of two 4-year projects which have produced some very interesting results. Firstly, they have identified needs, requirements and challenges faced by the O&G industry and secondly, they suggested a list of products that the EO downstream market should try to develop in order to fix these challenges covering exploration and drilling activities, transports and logistics, risk and disasters and finally environmental monitoring. Recently, ESA, EARSC and the international Association of oil and gas producers (IOGP) have developed new collaborations to bring together EO and O&G communities and stimulate the use of Copernicus. The quantification of Copernicus data contribution was based on a 12 European and Canadian firms sample. The value of Copernicus data for this O&G upstream industry sample was approximately 8.75 million EUR [16].

Health Copernicus supports public health authorities in monitoring health-related environmental phenomena and offering relevant information. One of the most critical factors affecting public health is air pollution. Copernicus supports pollution monitoring

through the provision of daily air quality predictions. Furthermore, Copernicus can also help in identifying areas prone to the emergence and spread of epidemics, which are heavily dependent on environmental factors such as water, sanitation, food and air quality. Another case study refers to Numtech which is a French start-up that specialises in the simulation of the atmosphere. One of its main activities is to develop value added services considering air quality. They started using Copernicus three years ago for one of their products, urban air, which can provide high resolution air quality for cities. CAMS is the only initiative providing information on long distance pollutant fluxes free and at a global level. Copernicus gave Numtech the chance to reach cities where a limited amount of data is available. Copernicus improved the precision of urban air by 60% and we can evaluate an improvement of 10% to 20% to more local pollutants. About 10% of the data used in urban air stem from Copernicus, to be more specific it represents most of the data used to analyse the background pollution. This fact makes Copernicus one of the key inputs of urban air. Numtech expects from 1 to 10 million EUR of annual revenues on the Env&You project, so Copernicus should generate from 100K EUR to 1M EUR of annual revenues for this project [18].

Transport represents a really important sector of the European economy, and ensures that passenger's safety is a priority for carriers as well as governments and local authorities. Environmental hazards, such as volcanic eruptions or sea-ice can disrupt transportations flows with considerable economic effects. In the case of maritime transport for example, the data considering currents that Copernicus is providing can support ship routing service. Also, safe navigation in Arctic regions can be improved by this information [19].

Urban planning, Europe's municipalities and regions deal with diverse planning challenges, such as waste management or exploitation of renewable energies, which can have a serious impact on citizen's quality of life. In order to face these challenges, the Copernicus program provides information on land use and land cover classification, urban growth and green areas. In addition, Copernicus can provide assistance on monitoring the stability of infrastructures, assess new construction sites or assess population density. Building radar is a German start-up which supplies verified construction sales globally, offering information like a construction site location, building size and other data considering construction projects. Building radar is based on earth observation data, to be more specific Copernicus and internet data processed through their algorithms. Building's radar estimation is that the world market for sales leads in the construction sector at more

than 70 billion EUR. Building radar enabled its clients to gain time, improve their turnover and their overall sales performance. The building's radar model is heavily dependent on Copernicus (60% of the satellite data they use). In addition, 40% of their clients make use of applications that are based on Copernicus data, the other 60% is basically data that rely solely on internet data processing. Given their monthly turnover is between 10 to 50K, it can be calculated that 4K to 20K EUR is attributable to Copernicus. Furthermore, building radar is planning for a turnover growth rate of 30%, the share of turnover that is attributable to Copernicus is expected to increase in the future with the release of Sentinel-2B data and the usage of Sentinel-1 [20].

Insurance and Disaster Management, Copernicus data can be useful to public authorities considering all the phases of disaster management, from preparedness and prevention to identifying risks and preventing loss of lives. Considering the response phase Copernicus supports civil protection operations by offering products such as maps that can identify the extent of the disaster (e.g., delineation of the flooded area), as well as the level of damage (e.g., number of destroyed buildings during in case of an earthquake). Earth observation can also be of great use in the insurance sector by validating, updating or calibrating risk models. Agroseguro, is responsible for the management of the agricultural insurance on the Spanish market, as representative of the shareholding insurance companies, insuring productions, such as crops, livestock, aquaculture and forestry. Insurance against lack of pastures was developed in 2001, based on the Normalised Difference Vegetation Index (NDVI). This index measures the amount and lushness of vegetation. Based on Spanish market size and the share of meadows and pastures in Spain with regards to Europe a broad prediction of the future market value for livestock index insurance in Europe is around 516M EUR. Copernicus can offer the NDVI index through Sentinel-2A, but there remains uncertainty considering the final market penetration of the Sentinels. Based on the assumption that the share of sentinels will be about 1/3 of the total by 2025, the Copernicus enabled revenues for the European index products market that can be estimated around 172M EUR [21].

4.6. The ERA-5 Land Hourly Dataset

The dataset we used 'ERA5-Land hourly data from 1950 to present' is a reanalysis dataset that allows us to observe the evolution of land variables over several decades. Reanalysis datasets combine model data with observations from all over the world into an integrated and stable dataset based on the laws of physics. ERA5-Land mainly uses atmospheric variables such as air temperature and air humidity. The temporal and spatial dimensions of this dataset, as well as the fixed grid used for the data distribution at any period of time makes it appropriate for forecasting applications [13]. The main variables that Copernicus uses are:

The **10m u-component of wind** and **10m v-component of wind**, which are the Eastward-moving and Northward-moving air's horizontal speed at a height of ten meters above the Earth's surface respectively.
The **2m dew point temperature**, which is the temperature at 2 metres above the earth at which the air would have to be cooled for saturation to occur.
2m temperature, which is the temperature of air at 2m above the surface of land, sea or in-land waters.
Evaporation from bare soil, open water surfaces excluding oceans, the top of canopy and vegetation transpiration, these four variables measure the amount of evaporation at the top of the land surface.

Forecast albedo, which measures the reflectivity of Earth's surface.

Seven variables related to lakes, which are **lake bottom temperature**, **lake ice depth**, **lake ice temperature**, **lake mix-layer depth**, **lake mix layer temperature**, **lake shape factor** and **lake total layer temperature**.

Leaf area index high and low vegetation.

Runoff, which measures the amount of water from rainfall, melting snow which stays stored in the soil.

Skin reservoir content, the amount of water in the vegetation canopy.

Skin temperature, which is the temperature of the surface of the Earth.

Nine variables related to snow which are the **snow albedo**, **snow cover**, **snow density**, **snow depth**, **snow depth water equivalent**, **snow evaporation**, **snowfall**, **snowmelt** and **temperature of snow layer**.

Soil temperature level 1,2,3 and 4 these variables show the temperature of the soil in different layers of the ECMWF integrated forecasting system.

Surface latent heat flux, which measures the exchange of latent heat with the surface through turbulent diffusion.

Surface net solar radiation, which measures the amount of solar (shortwave) radiation reaching the surface of the earth (both direct and diffuse) minus the amount reflected by the earth's surface.

Surface net thermal radiation at the surface.

Surface pressure, measures the pressure of the atmosphere on the surface of land (atmospheric pressure)

Surface sensible heat flux, which measures the transfer of heat between the Earth's surface and the atmosphere through the effects of turbulent air motion.

Total evaporation, which measures the total amount of water that has evaporated from the Earth's surface.

Total precipitation, which is the accumulated liquid and frozen water, including rain and snow that falls to the Earth's surface.

Volumetric soil water layer 1,2,3,4, which measures the volume of water in different soil layers of the ECMWF integrated forecasting system [13].

5. Methodology

This research follows five steps to achieve its purpose, firstly (1) specify the problem, (2) collect the data, (3) data preparation, (4) explore data, (5) create the forecasting model, and finally (6) explain the forecasting results.

(1) Specify the problem. In recent times, there has been notable growth in the traffic sector considering the number of vehicles on the roads of Athens, this growth can cause serious transportation and health problems, due to this fact we developed a model that can predict the number of cars on a specific road in the city of Athens after two hours. As a result, we could contribute to the mitigation of congestion as well as to the reduction of vehicle pollutants. Furthermore, Shapley additive values (SHAP) was used to explain the results.

(2) Collect the data. In order to collect our data, we used two open-source datasets. The first one was from https://data.gov.gr/, where data published by the central government, local authorities and public bodies are available. We downloaded road traffic data for the Attica region from a sensor that was placed on a specific road of Athens and was counting the average speed and the number of cars that were passing each hour every day for the first 6 months of 2022. In order to improve our prediction and see how much traffic can be affected by the different weather each day, we downloaded a second dataset from https://www.copernicus.eu/en, the European Union's Earth observation program. From the vast variety of variables that Copernicus had to offer we chose the ones that we believed were the most relevant with traffic, to be more specific our variables were eastward-moving air speed, northward moving air speed, dew point temperature, temperature, surface solar radiation and total precipitation.

(3) Data preparation. After data was possessed, the necessary pre-process was needed. Firstly, in order to merge our two datasets, we converted the datetime columns in both datasets to the appropriate format and then merged them by using datetime as a key. The next step was to clean and transform our merged dataframe from the unnecessary columns that gave us no further information about the number of cars that were recorded from our sensor. Finally, the remaining columns were renamed to help us better explain our final results.

(4) Explore data. During this step, a variety of visualisations were used to explore and understand the type of our time series and also search for underlying trends in our dataset that we might need to account for. Furthermore, the autocorrelation function (ACF) was used to examine the relationship between the number of cars in an hour and the same number during the past hours. The association between data and itself, lagged by a certain number of time units is displayed via autocorrelation, for instance lag 1 is a copy of the original data that has been time-shifted by one period [60].

(5) Create the forecasting model. In order to create our forecasting model, we used the Extreme Gradient Boosting (XGboost) algorithm. This algorithm is mainly based on gradient decision trees and uses a second-order Taylor expansion in order to calculate the loss function. It is an ensemble method that combines the outputs from individual decision trees. This algorithm has been applied to a wide range of fields, including health, finance and energy. In addition, we used cross-validation to tune our model. To be more specific, we used the TimeSeriesSplit method, which is a variation of the k-fold cross-validation. In the kth split, it uses the first k folds as a train set and the (k+1)th fold as a test set, while ignoring the future values in each split. We used five splits to split our dataset into 75% train data and 35% test data. To evaluate the performance of our model we used the mean absolute error (MAE), the root mean squared error (RMSE) and the mean absolute percentage error (MAPE) metrics.

(6) Explain the forecasting results. Along with accurately predicting the number of cars, it is crucial to understand the factors that affect our prediction. Therefore, we used the Shapley additive explanations (SHAP). The main idea behind SHAP is to calculate the Shapley values for each feature of the sample that has to be interpreted, where each value represents the impact of each feature to the final result. Additionally, to show each variable's contribution to each prediction of the model. More important variables appear higher in the summary plot.

6. Data Collection

Two datasets were collected in order to create a model that predicts and explains the traffic state for the city of Athens.

6.1. Traffic Dataset

The first dataset includes open-source data from https://data.gov.gr/.Governments, municipal governments, and other public entities make their Open Government Data (OGD) accessible to the general public through their data portals. The official data portals of governments frequently post traffic data produced by sensors as OGD. The official Greek data portal for OGD is called data.gov.gr. Data produced by the Greek national government, local governments, and other public entities are accessible through the data portal's new, updated edition, which was introduced in 2020 and is divided into ten topical categories (e.g., environment, economy, and transportation). Roadside sensors that broadcast information on the quantity and speed of moving vehicles are known as "traffic data." Since the data are pooled hourly, they have been sufficiently anonymised to prevent privacy concerns. In order to obtain this data, we used the data.gov.gr API (Application Programming Interface). An API is a set of rules that dictate how two machines communicate with each other, API call is actually the request between the user and the web server. Every time we use software to communicate with other software or online servers, we are using APIs to request the information we need. In our case we used data.gov.gr API to obtain data considering traffic in the roads of Athens, for every hour, each day during the first six months of 2022. Each row includes the exact time and date of the record, the sensor's id, the number of cars at this time, the name of the road, and the average speed of cars. In order to use the API service that gov.gr offers, users need to get a token by creating an account and providing personal information (such as name, email address and organisation) as well as the reason that they want to use the API.

6.2. Weather Dataset

The second dataset includes open-source data from https://www.copernicus.eu/en, to obtain our data we used the climate data store application program interface (CDS API). The Climate Data Store (CDS) API is a service that gives permission to the users to request data from CDS datasets by using a python script. These scripts contain a number of keywords, which can vary depending on the dataset that the user wants to retrieve data

from. The categories that these keywords belong to are product type such as ‘reanalysis’, variables of interest like ‘temperature’ or ‘total precipitation’, the year, month, day or even time that the data are retrieved for, the format that we want the data to be (‘netCDF’, ‘grib’) and the area that we want to extract data from [3]. In order to use the CDS API service we have to install the CDS API client which is a Python based library, by running the following command in a command prompt window: ‘pip install cdsapi ‘. Once the CDS API client is installed, it can be used in order to retrieve data from the available datasets in the CDS and ADS (Atmosphere Data Store) catalogues. Attached to each dataset download form there is the option of ‘Show API Request’ which displays the python code to be used [3]. The CDS contains a wealth of observations, climate data and seasonal forecasts. The result is a Time series dataset of on-site observations and reprocessed climate data records from satellites. To create a more accurate traffic forecasting model, from the 50 available variables that Copernicus has to offer, we chose 2m temperature, total precipitation, snowfall, dewpoint temperature, the Eastward-moving and Northward-moving air’s horizontal speed at a height of ten metres above the Earth’s surface and surface net solar radiation, hence we believe that these specific variables can affect the future traffic state of a road network.

7. Data Exploration

In this project we used two datasets, (i) traffic data from a sensor placed near Omonoia square, which is a very popular road in the city of Athens as well as, (ii) weather data for this specific from Copernicus**. Table 1** reports the descriptive statistics of the number of cars during each hour for each month near Omonoia square. Our starting period is the 1st of January 2022, according to our table we notice that there is not significant variation in the mean number of cars during each month, hence our data is stationary.

Table 1 Descriptive Statistics of The Number of Cars During Each Month

	january	February	March	April	May	June
count	712.00	670.00	737.00	713.00	744.00	94.00
mean	1,142.02	1,601.91	1,258.56	1,457.62	1,522.74	1,666.38
std	968.17	1,056.04	1,055.00	1,051.43	995.95	1,012.65
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	280.00	600.00	240.00	480.00	590.00	800.00
50%	920.00	1,580.00	1,080.00	1,320.00	1,520.00	1,640.00
75%	1,880.00	2,440.00	2,120.00	2,280.00	2,320.00	2,470.00
max	4,120.00	4,400.00	4,240.00	4,520.00	4,200.00	4,120.00

In addition, **Fig. 6** shows the hourly variation of the number of cars during the first six months of 2022. The figure doesn't reveal any kind of a strong trend during the examined period.

Considering the monthly variation of the number of cars during each month (**Fig 7**), we notice that in January we had the fewer number of cars and in February the highest. Overall, there is not significant variation to the number of cars during our examined period, nor does the boxplot reveal one.

Regarding the number of cars during each hour of the day (**Fig 8**), according to the plot, we can clearly see that small numbers are mainly noticed during the early hours of the day and after approximately five o'clock whereas, the highest can be noticed from approximately ten to four o'clock.

Finally, we checked our data for autocorrelation. Firstly, we performed a Durbin-Watson Test. The test produced a test statistic equal to 0.111, which indicates that there is autocorrelation in the data. This means that there is a significant degree of similarity between our given time series and a lagged version of itself over successive time intervals. In addition, we created the autocorrelation plot (**Fig 9**). The x-axis illustrates the lag values (1=previous hour, 2= 2 previous hours, etc). The y-axis indicates the amount of autocorrelation. According to our plot (**Fig 9**) the previous hour has the strongest impact considering the number of cars and as we move to the previous hours the degree of impact is decreasing.

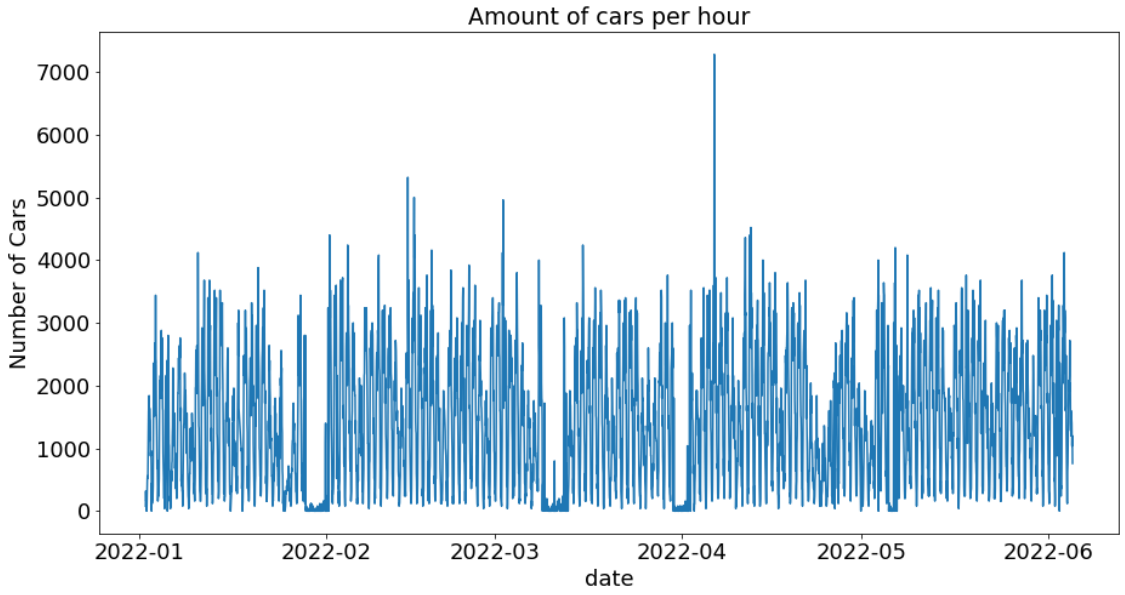


Figure 6 Number of Cars Each Hour of The Day During Six Months

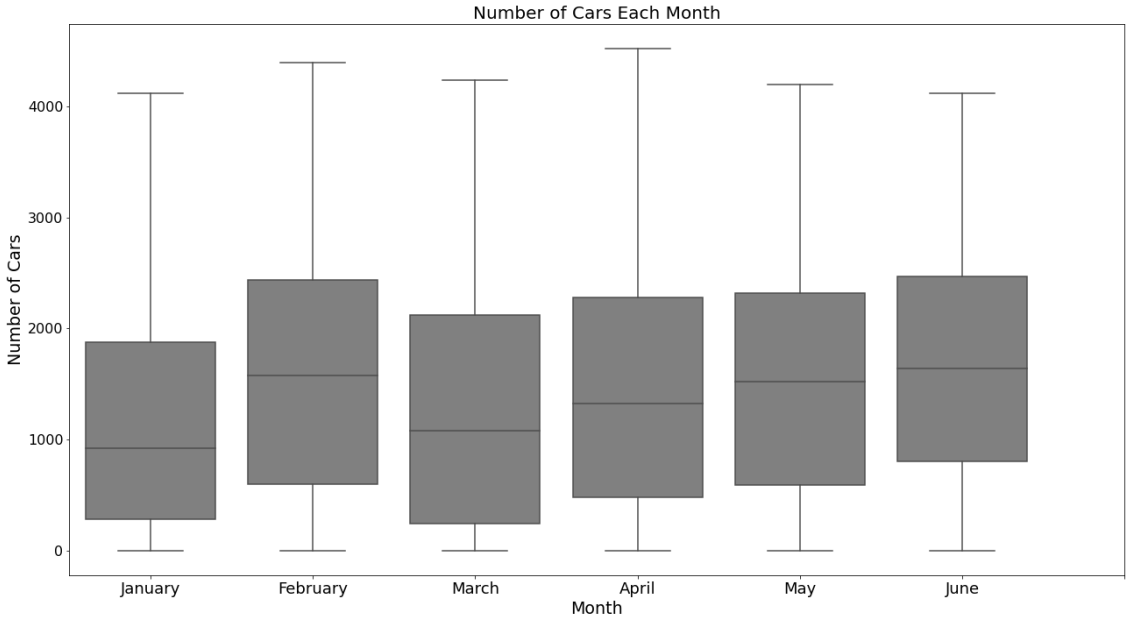


Figure 7 Variation in The Number of Cars During Each Month

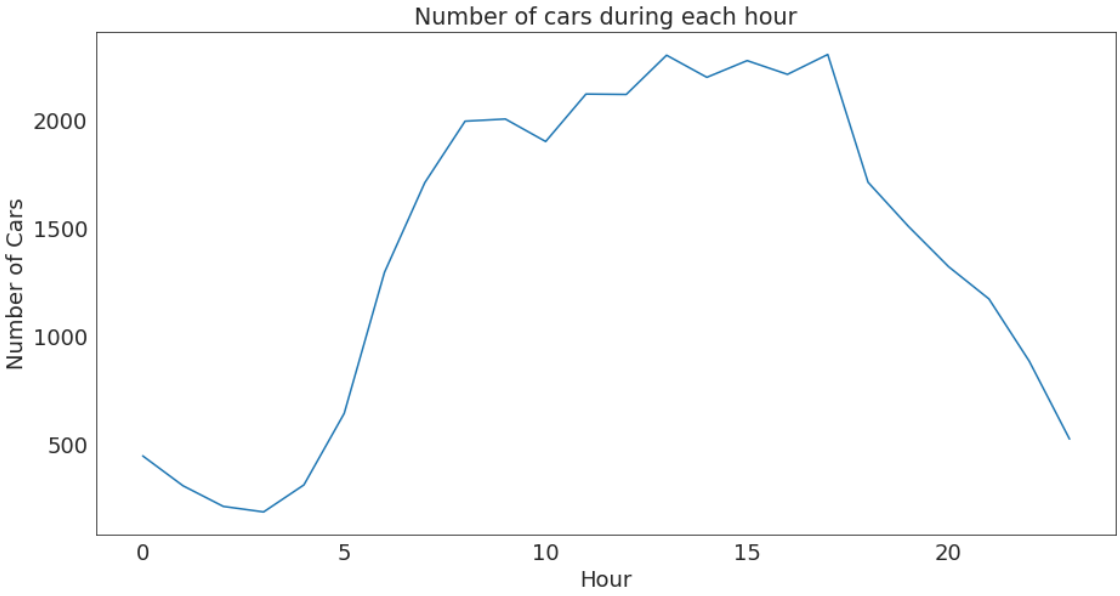


Figure 8 Number of Cars During Each Hour of The Day

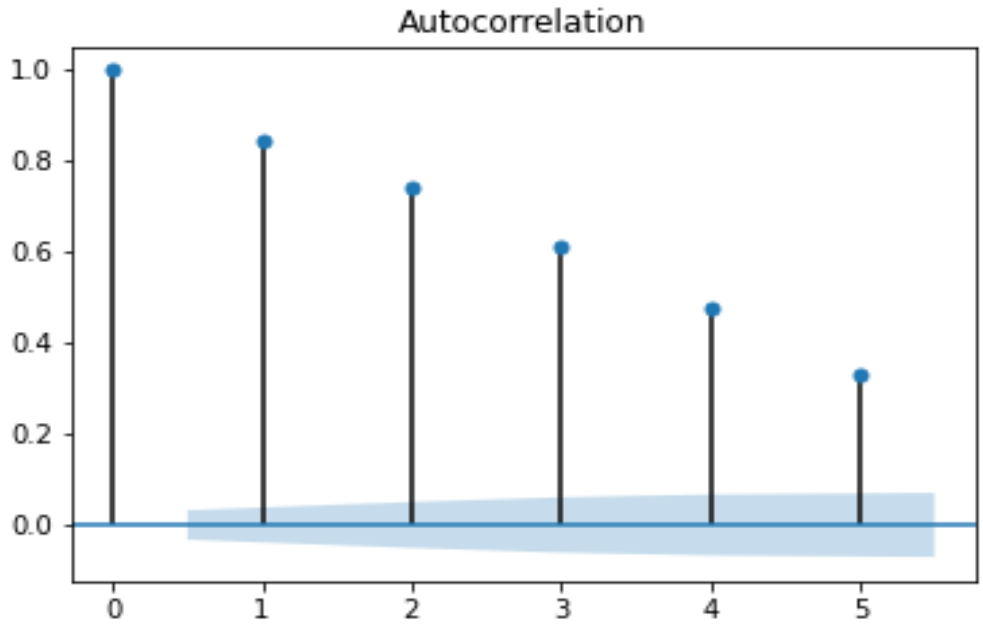


Figure 9 Autocorrelation Plot

8. Traffic Forecasting

To create our forecasting model, we used four types of variables (i) date-time variables, (ii) lag variables (variables from the past), (iii) weather related variables from and finally (iv) function variables.

Date-time-based variables are numeric features that are components of the time step itself for each observation in our dataset. They indicate the day, month, hour, weekend and daytime or night time (**Table 2**).

Table 2 Time-Based Variables

Variable	**Description**
1. Seg_hour	Which hour of the day the number of cars was recorded. Range between [0-23].
2. Day	Which day of the month the number of cars was recorded. Range between [1-31].
3. Month	Which month of the year the number of cars was recorded. Range between [1-6].
4. Day/Night	Returns if the number of cars was recorded during day or night. Returns 1 if it was recorded during day and 1 during night.
5. Weekday	Which day of the week the number of cars was recorded. Range between [0-6].

Lag variables represent values at prior timesteps that are considered useful because they are created on the assumption that what happened in the past can have an impact or contain a sort of intrinsic information about the future **(Table 3)**.

Table 3 Lag Variables

Variable	**Description**
1. Prev_2hours	The number of cars recorded 2 hours ago.
2. Prev_2hours_mean	The mean of the number of cars recorded 2 hours ago.
3. Prev_2hours_max	The max value of the cars recorded 2 hours ago.
4. Prev_2hours_min	The min value of the cars recorded 2 hours ago.
5. Prev_3hours_mean	The mean value of the cars recorded 3 hours ago.
6. Prev_3hours_max	The max value of the cars recorded 3 hours ago.
7. Prev_3hours_min	The min value of the cars recorded 3 hours ago.

Weather related variables are numeric variables that provide information about the temperature, the surface solar radiation and the total precipitation near Omonoia square, which is located in the city of Athens where our traffic sensor is placed **(Table 4)**.

Table 4 Weather Variables

Variable	**Description**
1. Eastward-moving air speed	It is the horizontal speed of air moving towards the east, at a height of ten metres above the surface of the Earth, in metres per second.
2. Northward-moving air speed	It is the horizontal speed of air moving towards the north, at a height of ten metres above the surface of the Earth, in metres per second.
3. Temperature	Temperature of air at 2m above the surface of land, sea or in-land waters.
4. Surface solar radiation	It is the Amount of solar radiation reaching the surface of the Earth minus the amount reflected by the earth's surface.
5. Total precipitation	It is the accumulated liquid and frozen water, including rain and snow, that falls to the earth's surface.

Finally, function variables are our last two features cos_seg_hour and sin_seg_hour. In order for our model to detect the cyclical pattern of the hour during the day, we used cos and sin functions. Firstly, we normalised our seg_hour variable to range between 0 and 2π, afterwards we implemented the cos and sin functions to get our two new features.

To create our model, we used the XGboost algorithm that predicts the number of cars after two hours during each day for six months on the road that we have placed our sensor on. In order to improve our algorithm's accuracy, we used cross validation to tune our model. This process is really important in any machine learning implementation, the reason is that it helps us select the optimal parameters for our machine learning algorithm

and as a result improve our model's accuracy. **Table 5** shows the optimal hyperparameters values of our prediction model

Table 5 Hyperparameters Values

Parameter	**Optimal Value**
1. n_estimators	1000
2. learning rate	0.01
3. colsample by tree	0.77
4. gamma	0
5. max depth	3

To effectively split our time series data, we created a custom class called "Blocking Time Series Split" using scikit-learn's GridSearchCV function to perform hyperparameter tuning on an XGBoost regressor model. using scikit-learn's GridSearchCV function to perform hyperparameter tuning on an XGBoost regressor model. The BlockingTimeSeriesSplit class defines a split method that generates indices for each split of the dataset. The split method takes in X, y, and groups as arguments, where X represents the input features, y represents the target variable, and groups represents a grouping variable. However, y and groups are not used in this implementation. The n_splits parameter specifies the number of splits to be generated. For each split, the method yields two arrays of indices: the first contains the indices for the training set, and the second contains the indices for the validation set. The training set consists of the first 70% of the data in each split, while the validation set consists of the remaining 30%. The XGBoost regressor model is instantiated, and GridSearchCV is used to perform hyperparameter tuning on the model. paramGrid should contain a dictionary of hyperparameters to be searched over. The cv parameter is set to an instance of the BlockingTimeSeriesSplit class with n_splits set to 5, indicating that the dataset will be split into 5 folds using the blocking time series cross-validation strategy. The scoring parameter is set to 'neg_mean_absolute_error', which specifies the scoring metric to be used in the cross-validation evaluation. The fit method of the GridSearchCV object is

called with the training data, X_train and y_train. After fitting, the best_score_ attribute of the fit object is printed to the console, which represents the best mean score achieved by the model over all cross-validation folds. The best_params_ attribute is also printed to the console, which represents the hyperparameters that achieved the best score.

To evaluate the accuracy of our prediction, we compared it with a baseline model. Specifically, we assumed that as we run our model the number of cars will remain the same as it was two hours before. Furthermore, we used three evaluation metrics MAE (mean absolute error), RMSE (root mean square error) and MAPE (mean absolute percentage error). Table 6 and Table 7 show the evaluation metrics values of our prediction and baseline models.

Table 6 Baseline Evaluation Metrics

	MAE	RMSE	MAPE (%)
Baseline	542.4	759.04	4.67

Table 7 Prediction Evaluation Metrics

	MAE	RMSE	MAPE (%)
Train	272.27	396.86	3.15
Test	325.12	474.77	4.29

Figure 10. Shows the difference between actual and predicted values in the test set. Based on the plot, it is noticeable that our model is able to accurately predict the number of cars near Omonoia square except for peak periods.

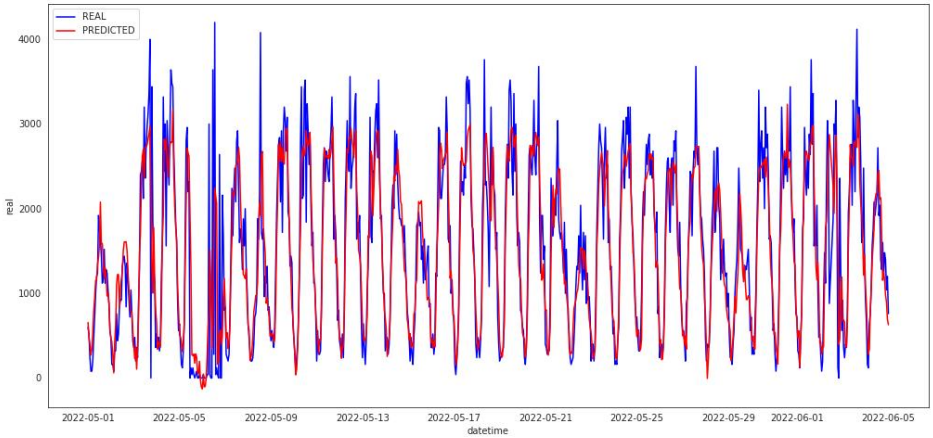


Figure 10 Comparison Plot

9. Model Explanation

In order to explain our model, we used the SHAP summary plot. **Figure 11** illustrates the summary plot. A SHAP (SHapley Additive exPlanations) summary plot is a graphical representation of the importance of each feature in a machine learning model. It helps to understand how each feature contributes to the model's predictions. The plot displays the mean absolute value of the SHAP values for each feature, sorted in descending order. The SHAP values represent the impact of a particular feature on the model's output. A positive SHAP value means that the feature has a positive impact on the model's output, while a negative SHAP value indicates a negative impact. The mean absolute value of the SHAP values for a feature is a measure of the feature's overall importance in the model. Each observation of our dataset is represented as a dot, in our case each dot represents one hour during the day. The model's input variables are displayed on the y-axis in decreasing order of significance (from top to bottom). In the plot, variable 'prev_two_hours_mean' has the highest impact on the number of cars during each hour, followed by 'cos_seg_hour', which is the hour of the day after we normalised it between 0 and 2π and then used the cos function. The third variable that has the highest effect on our result is 'prev_two_hours_max' which is the hour closer to 23:00 o'clock from the previous two hours. Furthermore, in a summary plot of SHAP values, the x-axis typically represents the feature or variable being explained. A feature with a high SHAP value has a large impact on the model's prediction for a particular instance. This means that the model's prediction changes significantly when that feature is included or excluded. A feature with a high SHAP value is therefore considered to be an important contributor to the model's prediction for that instance. The colour of the dots reveals if a variable has high (red), or low (blue) value for each observation. According to our plot, if we observe the 'cos_seg_hour' variable we notice that during early hours (blue colour), has higher SHAP value meaning that at this hour the probability of having a big number of cars on our road is high. On the other hand, during late hours the SHAP value is lower meaning that the same probability is low.

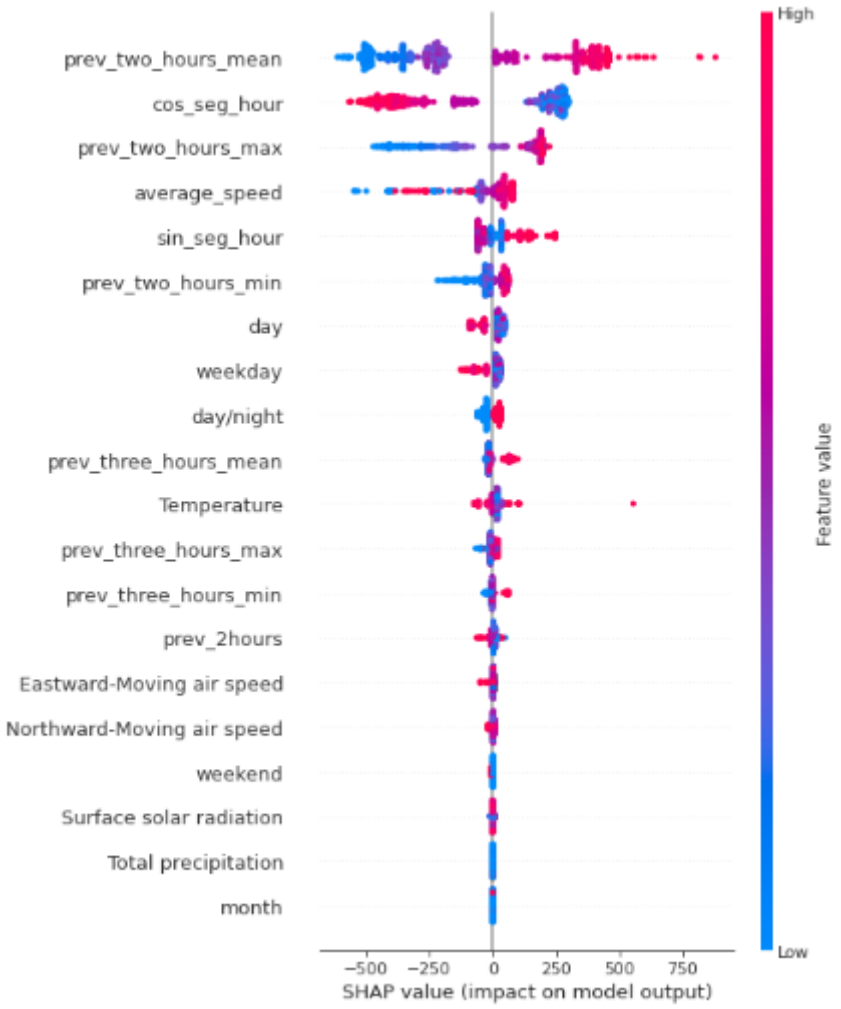


Figure 11 Summary Plot

Figure 12, shows a Local bar plot which is an essential visualization tool that plays a crucial role in interpreting the feature importance of machine learning models, particularly in the context of explaining individual predictions made by the model. The plot provides us with a clear representation of the contribution of each feature to the predicted output for a specific instance of input data, allowing us to gain deeper insights into the behavior of the model. In the Local bar plot, we can see a horizontal bar for each feature, with the length of the bar representing the magnitude of the SHAP value. The SHAP (Shapley Additive exPlanations) value is a statistical measure that quantifies the impact of each feature on the model's output. The colour of the bar indicates whether the feature value is high (red) or low (blue) relative to the other instances of data in the dataset. This provides us with a clear understanding of how each feature compares to other instances in the dataset, making it easier to interpret the results. Upon analyzing our

Local bar plot, we can observe that the prev_two_hours_mean and prev_two_hours_max features hold the most substantial impact on our target variable, with both of their values being relatively low compared to other instances in the dataset. This indicates that the model is highly sensitive to these features, and any changes to their values can have a significant impact on the predicted output. In contrast, the Sin_seg_hour feature, which is the third most important variable in our plot, has a relatively high value compared to other instances in the dataset. This suggests that the model is less sensitive to this feature and that changes to its value may have a lesser impact on the predicted output.

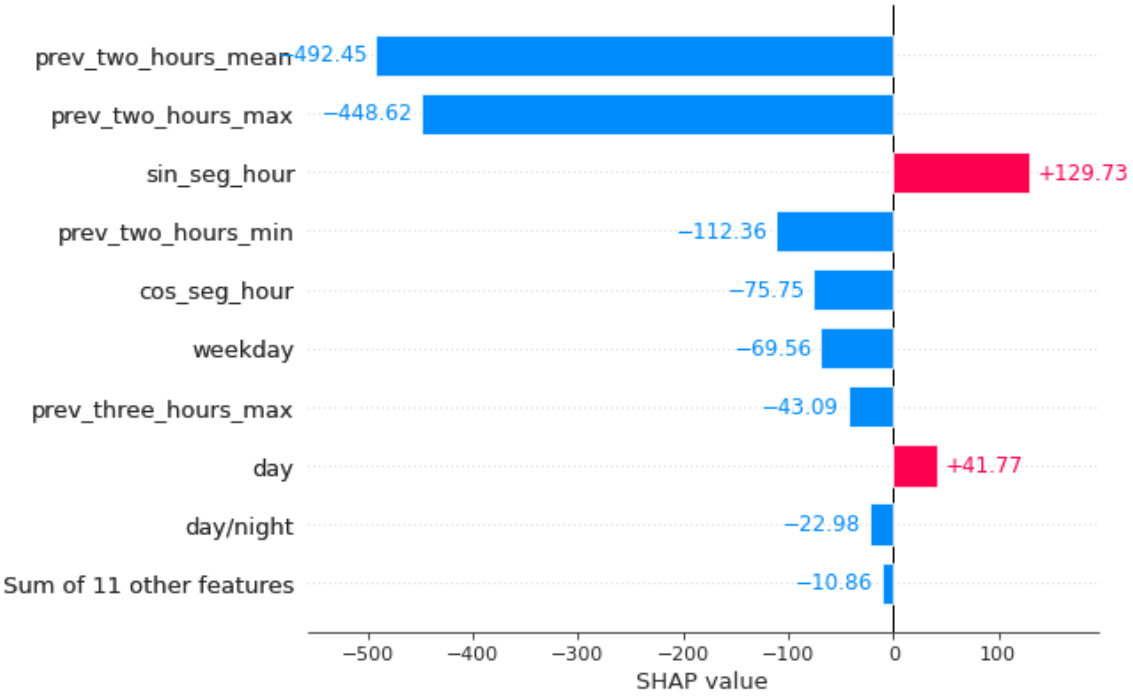


Figure 12 Local Bar Plot 1

In Figure 13, we can see another instance of our dataset displayed in a local bar plot. The plot reveals that prev_two_hours_mean and cos_seg_hour have the greatest impact on our target variable, indicating that any changes made to their values will significantly affect the target variable. In comparison to our first example, we can observe that the feature values are relatively high (represented in red), unlike other instances in the dataset. This indicates that these features have a strong positive impact. Specifically, the variable prev_two_hours_mean, which was the most critical variable in our previous example, now has a positive impact on our prediction instead of a negative one. This demonstrates that different instances in the dataset can have varying feature importance and impact on our prediction, highlighting the importance of interpretability and explainability in machine learning models.

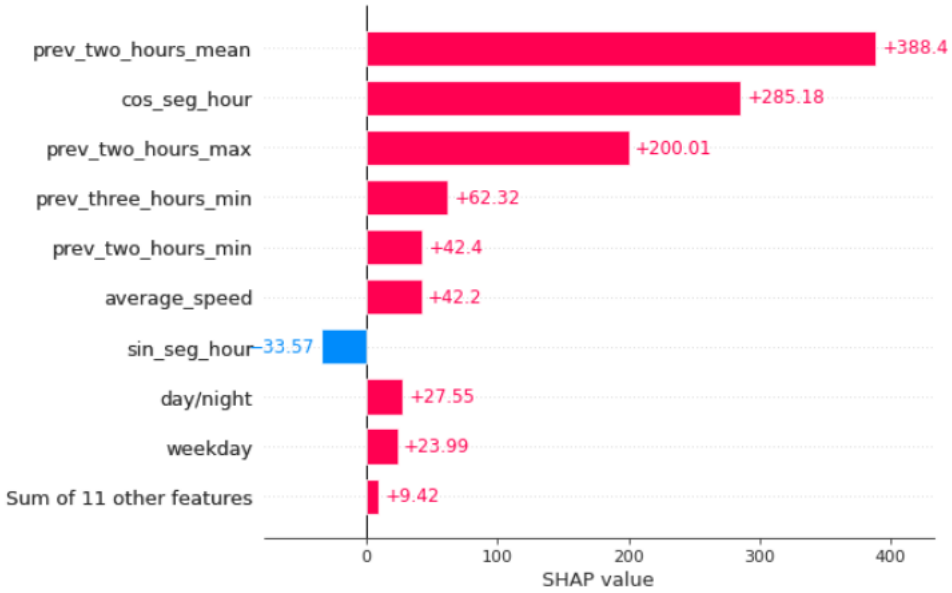


Figure 13 Local Bar Plot 2

Figure 14, Illustrates the feature importance bar plot which is a critical visualization tool, used to evaluate the predictive power of different features on the target variable in a model. By analyzing the relative importance of each feature, we can identify which variables have the most significant impact on the outcome of our model. In the feature importance bar plot, the y-axis represents the names of the most important features of the model, while the x-axis indicates the feature importance scores. This bar plot provides a quick and easy-to-read visualization of how each feature contributes to the overall performance of the model. Upon analyzing the feature importance bar plot for our model,

we can observe that the previous three hours mean feature holds the most substantial impact in predicting the number of cars after two hours. It is followed by the previous two hours min variable, previous two hours max, and cos_seg_hour. These variables hold considerable importance in the model and contribute significantly to the prediction of the target variable. By leveraging the insights provided by the feature importance bar plot, we can optimize our model by focusing on the most important variables and improving their predictive power. This will help us build more accurate and reliable models that can provide us with valuable insights into the behavior of our target variable.

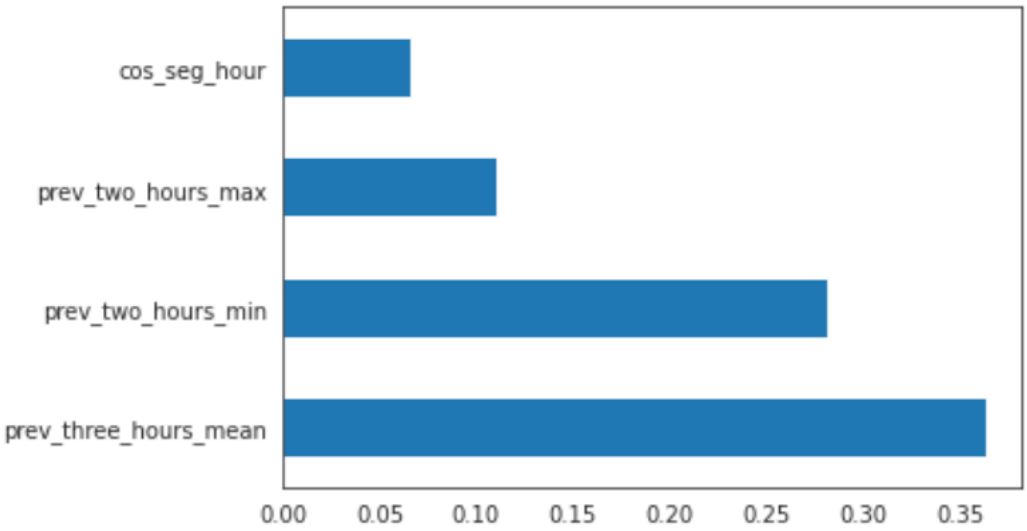


Figure 14 Feature Importance Bar Char

10 Conclusion

The main objective of this study was to use machine learning and XAI (Explainable artificial intelligence) to predict and explain the traffic state of a road network. As a result, a case study was presented that uses the XGBoost algorithm to predict the amount of traffic on a specific road of Athens during the next two hours. Furthermore, the SHAP framework was used to explain the model and help the user understand the forecasts that were made. Moreover, in order to improve the accuracy of our model weather data was added from Copernicus, the European Union's Earth observation program [23]. 26 variables were used to create the model, which were divided into three groups: time-based factors, weather variables and function variables. The procedure of our model evaluation resulted in an RMSE value of 474.77, an MAE value of 325.12, an MAPE value of 4.29. The model outperformed the baseline model whose MAE value was 542.4 which is significantly more than our model's. Furthermore, by using the SHAP framework we found that the 'prev_two_hours_mean', the 'cos_seg_hour', the 'prev_two_hours_max' and the average speed were the variables that had the biggest impact on our prediction for the number of cars during the after two hours. In addition, we found out that during the early hours of the day there was a greater likelihood of having a lot of cars on our route. The implementation of efficient traffic control measures is required due to the numerous major transportation and health issues generated by the ever-growing volume of road traffic. Traffic forecasting plays a crucial role in improving traffic management by providing real-time and accurate predictions of traffic flow on roads and highways. This information can then be used by traffic management authorities to make informed decisions about traffic control, such as adjusting traffic signals, rerouting traffic, and deploying emergency resources. Real-time traffic prediction. By providing real-time predictions of traffic flow, traffic management authorities can quickly respond to changing traffic conditions and make adjustments to traffic control systems as needed. Incident management. By providing early warning of traffic incidents, such as accidents or road closures, traffic management authorities can quickly deploy emergency resources and minimise the impact on traffic flow. Congestion management. By predicting areas of potential congestion, traffic management authorities can take proactive measures to prevent or alleviate congestion, such as adjusting traffic signals or rerouting traffic. Long-term planning. By analysing historical traffic data, traffic management authorities can

make informed decisions about long-term infrastructure projects, such as the construction of new roads or the expansion of existing ones. Overall, traffic forecasting can play a critical role in improving traffic management by providing real-time and accurate predictions of traffic flow, enabling traffic management authorities to make informed decisions and take proactive measures to improve traffic flow and safety.

11. Appendix

Importing Libraries

```
!pip install shap
!pip install sklearn metrics
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgb
from sklearn.model_selection import TimeSeriesSplit
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error
import shap
from statsmodels.graphics import tsaplots
from statsmodels.stats.stattools import durbin_watson
import math
```

First dataset gov.gr sensor ms121

```
df=pd.read_csv('/content/drive/MyDrive/data.xls')
sensore=df.loc[df['deviceid']=='MS121']
sensore.head()
```

	Unnamed: 0	deviceid	countedcars	appprocesstime	road_name	road_info	average_speed
260	260	MS121	240	2020-11-05T00:00:00Z	Λ. ΚΗΦΙΣΟΥ	ΡΑΜΠΑ ΕΙΣΟΔΟΥ ΑΠΟ ΑΓ. ΙΩ. ΡΕΝΤΗ (ΟΔΟΣ ΑΝΔΡΟΥΤΣ...	37.000000
446	446	MS121	40	2020-11-05T01:00:00Z	Λ. ΚΗΦΙΣΟΥ	ΡΑΜΠΑ ΕΙΣΟΔΟΥ ΑΠΟ ΑΓ. ΙΩ. ΡΕΝΤΗ (ΟΔΟΣ ΑΝΔΡΟΥΤΣ...	37.000000
867	867	MS121	200	2020-11-05T02:00:00Z	Λ. ΚΗΦΙΣΟΥ	ΡΑΜΠΑ ΕΙΣΟΔΟΥ ΑΠΟ ΑΓ. ΙΩ. ΡΕΝΤΗ (ΟΔΟΣ ΑΝΔΡΟΥΤΣ...	40.200000
1231	1231	MS121	120	2020-11-05T03:00:00Z	Λ. ΚΗΦΙΣΟΥ	ΡΑΜΠΑ ΕΙΣΟΔΟΥ ΑΠΟ ΑΓ. ΙΩ. ΡΕΝΤΗ (ΟΔΟΣ ΑΝΔΡΟΥΤΣ...	42.333333
1551	1551	MS121	440	2020-11-05T04:00:00Z	Λ. ΚΗΦΙΣΟΥ	ΡΑΜΠΑ ΕΙΣΟΔΟΥ ΑΠΟ ΑΓ. ΙΩ. ΡΕΝΤΗ (ΟΔΟΣ ΑΝΔΡΟΥΤΣ...	44.272727

Importing second dataset form Copernicus

```
weather_data=pd.read_csv('/content/drive/MyDrive/ms121f.csv')
weather_data['datetime']=pd.to_datetime(weather_data['time'],utc=True)
weather_data.head()
```

	longitude	latitude	time	u10	v10	t2m	ssr	tp	datetime
0	23.676001	37.962002	01/01/2022 00:00	0.924551	-2.309829	279.69427	4227250.0	0.000025	2022-01-01 00:00:00+00:00
1	23.676001	37.962002	01/01/2022 01:00	0.831941	-2.158489	279.28370	0.0	0.000000	2022-01-01 01:00:00+00:00
2	23.676001	37.962002	01/01/2022 02:00	0.747851	-1.851149	278.85815	0.0	0.000000	2022-01-01 02:00:00+00:00
3	23.676001	37.962002	01/01/2022 03:00	0.766709	-1.719187	278.71710	0.0	0.000000	2022-01-01 03:00:00+00:00
4	23.676001	37.962002	01/01/2022 04:00	0.743800	-1.768489	278.57068	0.0	0.000000	2022-01-01 04:00:00+00:00

Merging the two datasets

```
data=pd.merge(sensore,weather_data,on='datetime',how='left')
data.head()
```

	level_0	index	countedcars	average_speed	datetime	longitude	latitude	time	Eastward-Moving air speed	Northward-Moving air speed	Temperature	Surface solar radiation	Total precipitation
9099	9099	9099	1480	42.675676	2022-06-04 20:00:00+00:00	23.676001	37.962002	06/04/2022 20:00	-0.049882	1.674061	286.83524	18971484.0	0.000001
9100	9100	9100	1400	42.000000	2022-06-04 21:00:00+00:00	23.676001	37.962002	06/04/2022 21:00	-0.379675	1.445702	286.26624	18971484.0	0.000005
9101	9101	9101	1040	40.346154	2022-06-04 22:00:00+00:00	23.676001	37.962002	06/04/2022 22:00	-0.090110	2.100853	286.04453	18971484.0	0.000051
9102	9102	9102	1200	39.533333	2022-06-04 23:00:00+00:00	23.676001	37.962002	06/04/2022 23:00	-0.395180	1.678966	285.63210	18971484.0	0.000070
9103	9103	9103	760	39.631579	2022-06-05 00:00:00+00:00	23.676001	37.962002	06/05/2022 00:00	-0.816186	-0.923243	287.36655	20757920.0	0.000003

Creating new features seq hour, day, month, weekday, year, weekend,day/night, winter/spring

```
data['seg_hour']=data['datetime'].dt.hour
data['day']=data['datetime'].dt.day
data['month']=data['datetime'].dt.month
data['weekday']=data['datetime'].dt.weekday
data['year']=data['datetime'].dt.year
data['weekend']=[1 if data['weekday'][i] in [5,6] else 0 for i in rang
e(len(data))]
data['day/night']=[1 if data['seg_hour'][i]<18 and data['seg_hour'][i]
>6 else 0 for i in range(len(data))] # day equals 1 and night equals 0
data=data.drop(['time','longitude','latitude'],axis=1)
```

cyclical features encoding(cos-sin)

```
data['seg_hour_norm']=2 * math.pi *
data['seg_hour']/data['seg_hour'].max()
data['cos_seg_hour']=np.cos(data['seg_hour_norm'])
data['sin_seg_hour']=np.sin(data['seg_hour_norm'])
```

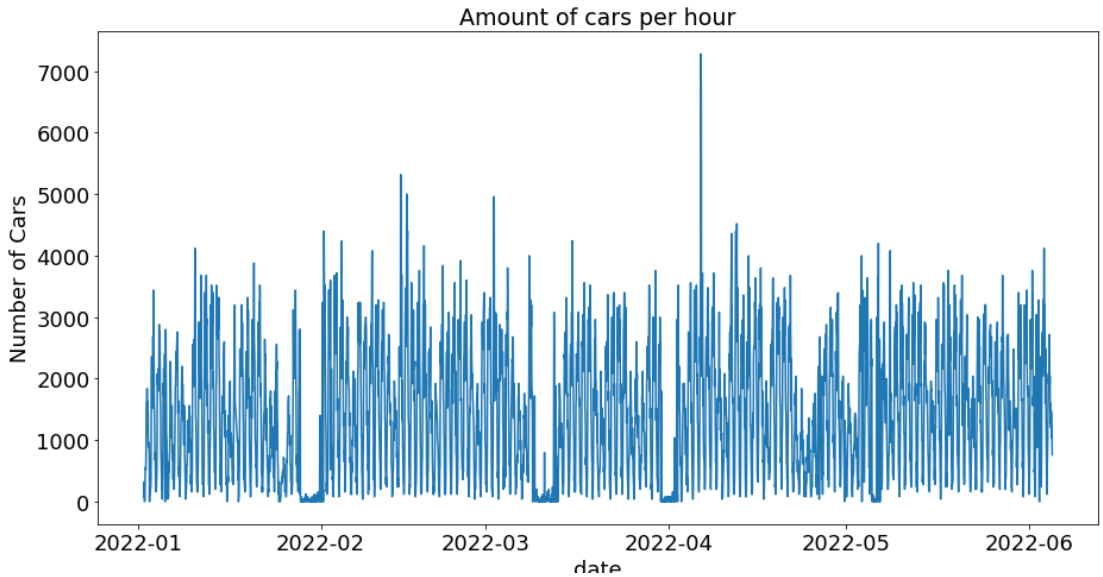
Descriptive statistics each month countedcars

```
data_pivot=data.pivot(index='datetime',columns='month',values='counted
cars')
data_pivot.columns=['january', 'February','March',
'April','May','June']
data_pivot.head()
data_pivot.describe().applymap('{:,.2f}'.format)
```

	january	February	March	April	May	June
count	712.00	672.00	738.00	715.00	744.00	94.00
mean	1,142.02	1,612.50	1,263.58	1,472.62	1,522.74	1,666.38
std	968.17	1,072.19	1,063.05	1,087.81	995.95	1,012.65
min	0.00	0.00	0.00	0.00	0.00	0.00
25%	280.00	600.00	240.00	480.00	590.00	800.00
50%	920.00	1,600.00	1,080.00	1,320.00	1,520.00	1,640.00
75%	1,880.00	2,440.00	2,150.00	2,280.00	2,320.00	2,470.00
max	4,120.00	5,320.00	4,960.00	7,280.00	4,200.00	4,120.00

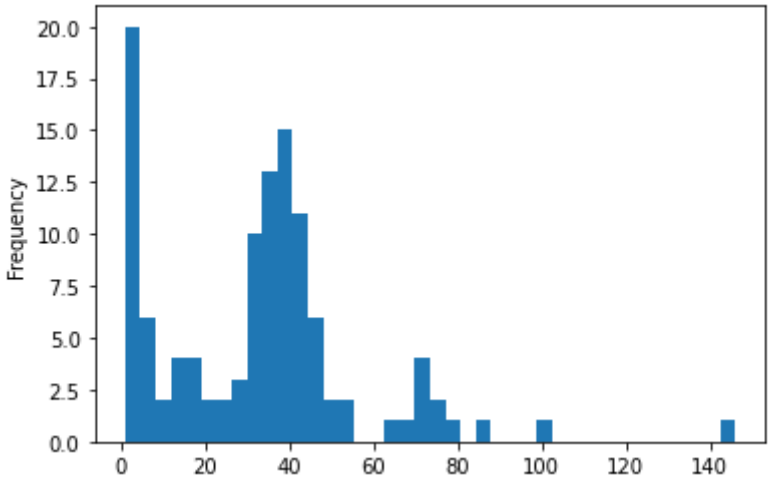
Number of cars per hour lineplot

```
fig=plt.figure(figsize=(13,7))
sns.lineplot(data=data,x='datetime',y='countedcars',ci=None)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.title('Amount of cars per hour',fontsize=19)
plt.xlabel('date',fontsize=18)
plt.ylabel('Number of Cars',fontsize=18)
plt.tight_layout()
plt.show()
```



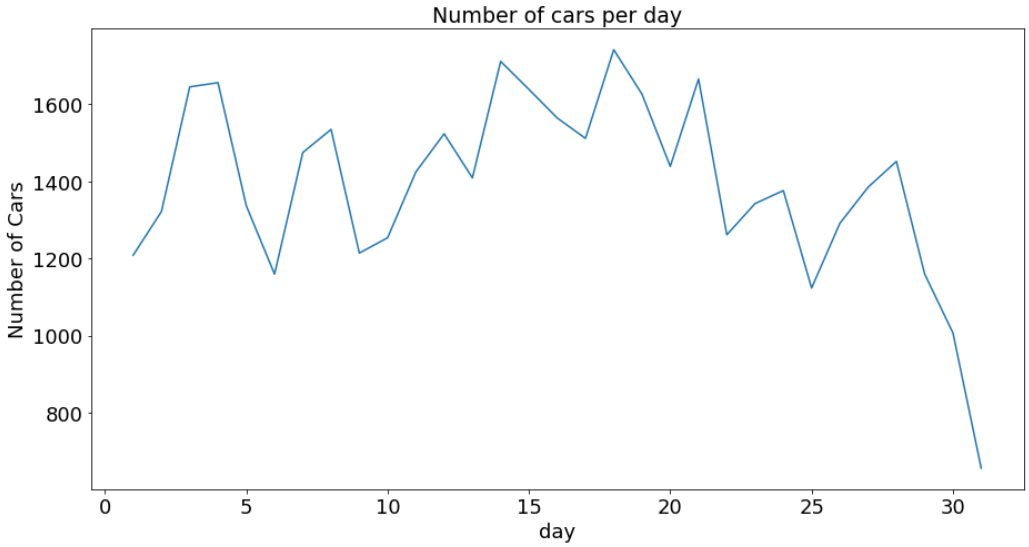
Countedcars(target variable) distribution

```
data['countedcars'].value_counts()\
.plot(kind='hist',bins=40)
```



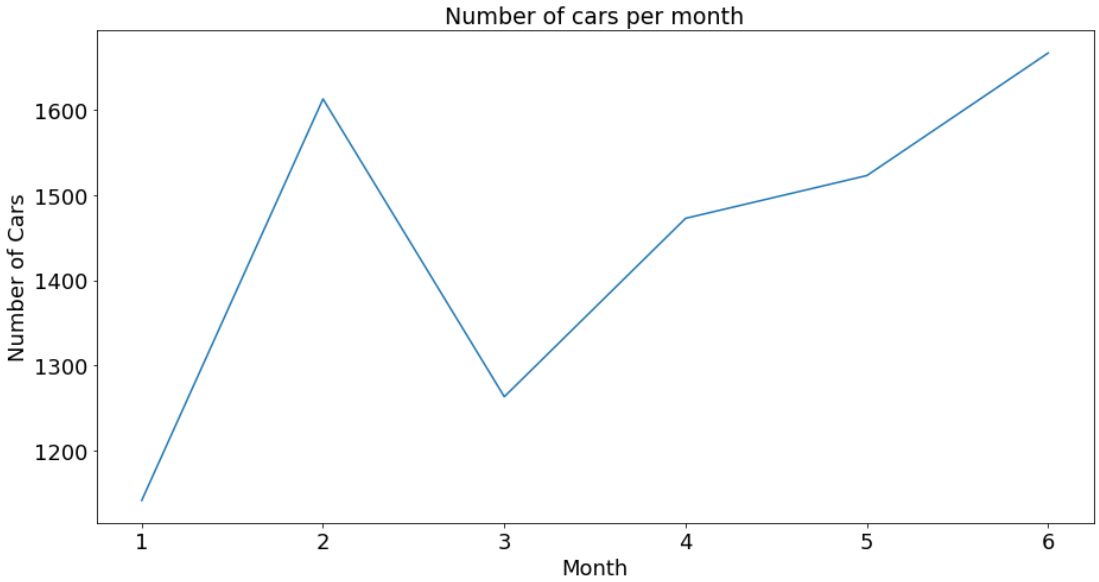
Number of cars during each day lineplot

```
fig=plt.figure(figsize=(13,7))
sns.lineplot(data=data,x='day',y='countedcars',ci=None)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.title('Number of cars per day',fontsize=19)
plt.xlabel('day',fontsize=18)
plt.ylabel('Number of Cars',fontsize=18)
plt.tight_layout()
plt.show()
```



Number of cars during each hour lineplot

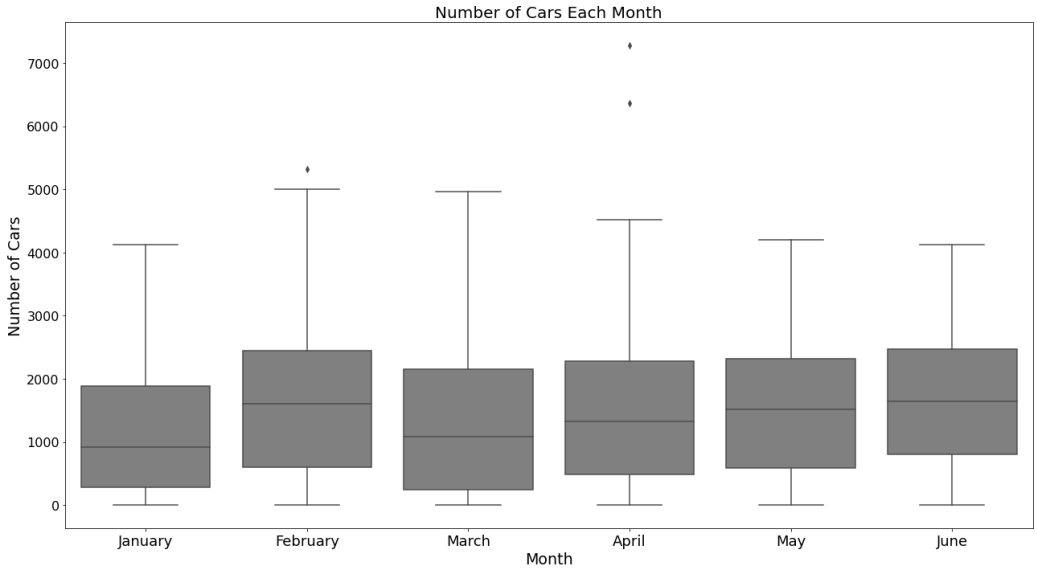
```
fig=plt.figure(figsize=(13,7))
sns.lineplot(data=data,x='month',y='countedcars',ci=None)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.title('Number of cars per month',fontsize=19)
plt.xlabel('Month',fontsize=18)
plt.ylabel('Number of Cars',fontsize=18)
plt.tight_layout()
plt.show()
```



Number of cars each month boxplot

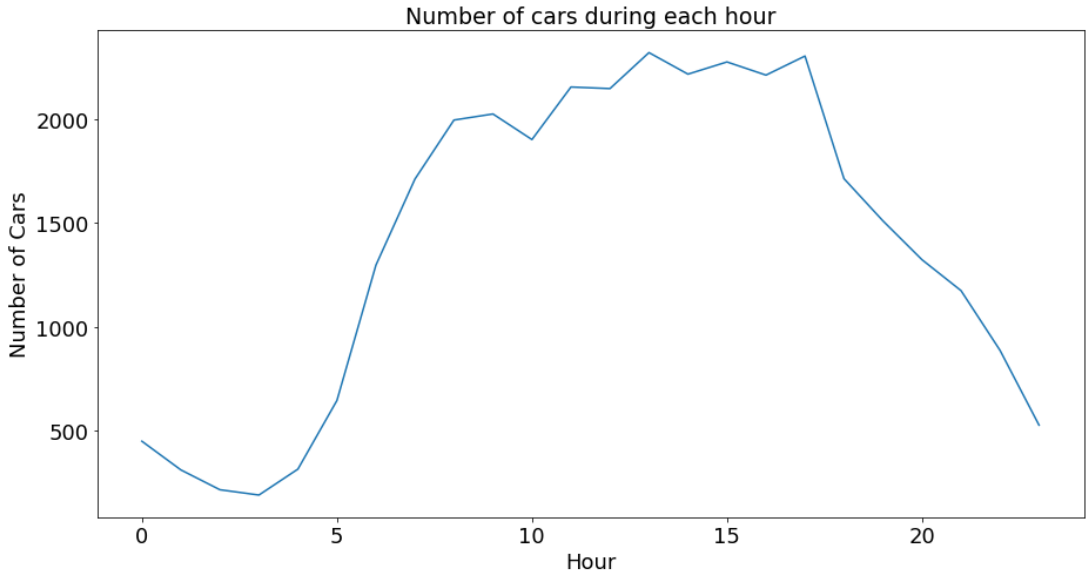
```
fig=plt.figure(figsize=(18,10))
sns.boxplot(data=data,x='month',y='countedcars',color='grey')
```

```
plt.title('Number of Cars Each Month',fontsize=20)
plt.xlabel('Month',fontsize=19)
plt.xticks(np.arange(6),labels=['January','February','March','April','
May',"June"],fontsize=18)
plt.yticks(fontsize=16)
plt.ylabel('Number of Cars',fontsize=19)
plt.tight_layout()
plt.show()
```



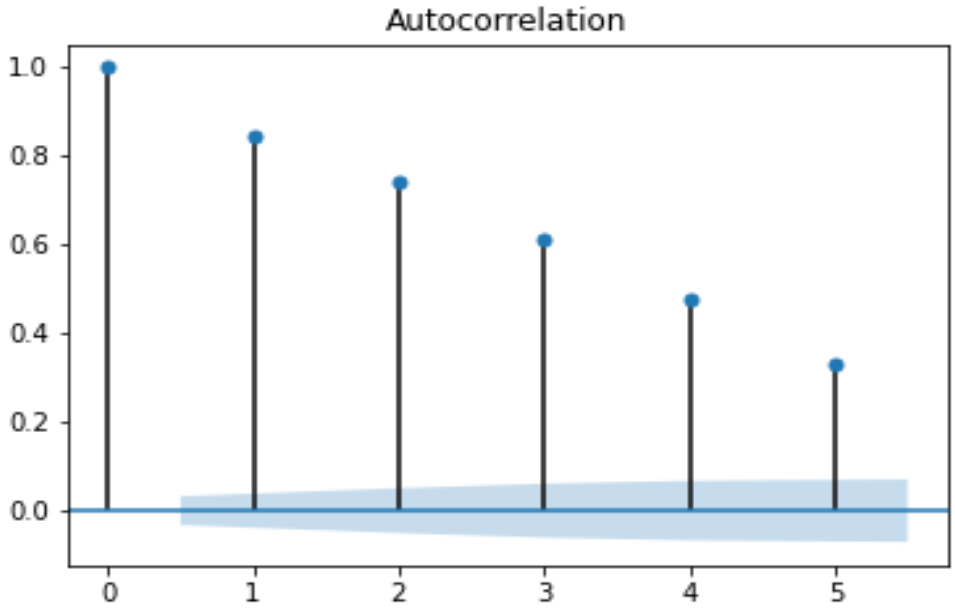
Number of cars during each hour lineplot

```
fig=plt.figure(figsize=(13,7))
sns.lineplot(data=data,x='seg_hour',y='countedcars',ci=None)
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
plt.title('Number of cars during each hour',fontsize=19)
plt.xlabel('Hour',fontsize=18)
plt.ylabel('Number of Cars',fontsize=18)
plt.tight_layout()
plt.show()
```



Autocorrelation Plot

```
fig = tsaplots.plot_acf(
    data["countedcars"], lags=5
)
plt.show()
```



Creating Lag features

```
data['index']=data.index
def prev_2hours():
   data['prev_2hours']=[data['countedcars'][i] if i==0 else data['coun
tedcars'][i-1] if i==1 else data['countedcars'][i-2]
                                for i in range(len(data))]
prev_2hours()
def prev_two_hours_mean(): #mean of previous two  days (window = 2)
      data['prev_two_hours_mean']=data['index'].apply(lambda x: data['
countedcars'].loc[x-2:x-1].mean()).round()

prev_two_hours_mean()

def prev_two_hours_min(): #minimum of  previous two on  days (window =
 2)
    data['prev_two_hours_min']=data['index'].apply(lambda x: data['cou
ntedcars'].loc[x-2:x-1].min())
prev_two_hours_min()

def prev_two_hours_max(): #maximum of previous two  days (window = 2)
     data['prev_two_hours_max']=data['index'].apply(lambda x: data['co
untedcars'].loc[x-2:x-1].max())

prev_two_hours_max()

def prev_three_hours_mean(): #mean of previous three  days (window = 3
)
     data['prev_three_hours_mean']=data['index'].apply(lambda x: data[
'countedcars'].loc[x-3:x-1].mean()).round()

prev_three_hours_mean()

def prev_three_hours_min(): #minimum of previous three  days (window =
 3)
     data['prev_three_hours_min']=data['index'].apply(lambda x: data['
countedcars'].loc[x-3:x-1].min())

prev_three_hours_min()

def prev_three_hours_max(): #maximum of previous three days (window =
3)
```

```
    data['prev_three_hours_max']=data['index'].apply(lambda x: data['
countedcars'].loc[x-3:x-1].max())

prev_three_hours_max()
```

Train-Test split (75-25)

```
train=data.loc[data.datetime < '2022-05-01 00:00:00+00:00']
test=data.loc[data.datetime >= '2022-05-01 00:00:00+00:00']
print(data.shape)
print(train.shape)
print(test.shape)
```

(3675, 26)
(2837, 26)
(838, 26)

Baseline model creation (Evaluation metrics: RMSE, MAE)

```
Baseline_pred=[data['countedcars'][i] if i==0 else
data['countedcars'][i-1] if i==1 else data['countedcars'][i-2]
for i in range(len(data))]
data_base=data['countedcars']
rmse_base=np.sqrt(mean_squared_error(data_base,Baseline_pred))
mae_base=mean_absolute_error(data_base,Baseline_pred)
mape_base=mean_absolute_percentage_error(data_base,Baseline_pred)
print("RMSE_base:",rmse_base,
"MAE_base:",mae_base,"MAPE_base",mape_base
```

RMSE_base: 759.0421461235447 MAE_base: 542.4 MAPE_base 4.6714889196017224e+16

Features-Target Separation

```
features=['Eastward-Moving air speed', 'Northward-Moving air speed',
      'Temperature', 'Surface solar radiation',
       'Total precipitation', 'average_speed',
       'cos_seg_hour','sin_seg_hour', 'day', 'month', 'weekday', 'week
end','prev_2hours',
       'prev_two_hours_mean', 'prev_two_hours_min', 'prev_two_hours_ma
x',
       'prev_three_hours_mean', 'prev_three_hours_min', 'prev_three_ho
urs_max','day/night']
target='countedcars'
```

X_train,y_Train,X_test,y_test creation

```
X_train=train[features]
y_train=train[target]
X_test=test[features]
y_test=test[target]
```

Parameter Tuning

```
paramGrid = {'n_estimators':[998],
             'learning_rate':[0.01],'max_depth':[2],
             'colsample_bytree':[0.77],
             'gamma':[0]
                }
```

Creating class BlockingTimeSeriesSplit to Split Time Series Data into Training and Test Sets

```
class BlockingTimeSeriesSplit():
    def __init__(self, n_splits):
       self.n_splits = n_splits

    def get_n_splits(self, X, y, groups):
        return self.n_splits

    def split(self, X, y=None, groups=None):
        n_samples = len(X)
        k_fold_size = n_samples // self.n_splits
        indices = np.arange(n_samples)

        margin = 0
        for i in range(self.n_splits):
            start = i * k_fold_size
            stop = start + k_fold_size
            mid = int(0.7 * (stop - start)) + start
            yield indices[start: mid], indices[mid + margin: stop]

btscv = BlockingTimeSeriesSplit(n_splits=5)
model=xgb.XGBRegressor()
```

```
gridsearch=GridSearchCV(model,paramGrid, cv=btscv,
scoring='neg_mean_absolute_error', verbose=1)

fit = gridsearch.fit(X_train, y_train)

print(fit.best_score_)

print(fit.best_params_)
```

Model Training

```
model=fit.best_estimator_
model.fit(X_train,y_train)
```

Model Prediction

```
y_pred_train=model.predict(X_train).round()
```

Evaluation Metrics

```
mae_train=mean_absolute_error(y_train,y_pred_train)
mae_train
272.2717659499471

mae_test=mean_absolute_error(y_test,y_pred)
mae_test
325.1252983293556

rmse_test=np.sqrt(mean_squared_error(y_test,y_pred))
rmse_test
474.77241501802996

mape_train=mean_absolute_percentage_error(y_train,y_pred_train)
mape_train
3.1512497780377764e+16
mape_test=mean_absolute_percentage_error(y_test,y_pred)
mape_test 4.295617162478804e+16
```

Shap Explainer Creation

```
best=model
explainer = shap.TreeExplainer(best)
```

Fits the explainer

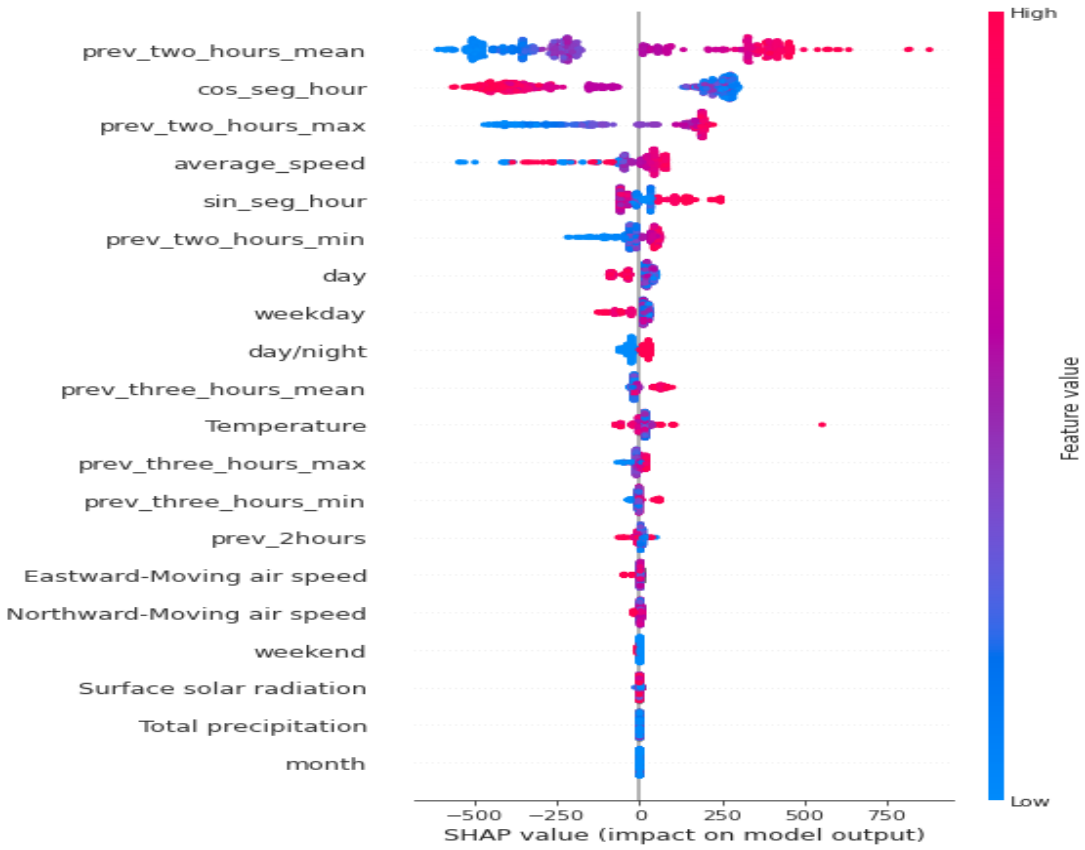
```
explainer = shap.Explainer(model.predict, X_test)
```

Shap Values Calculation

```
shap_values = explainer(X_test)
```

Shap values Plot

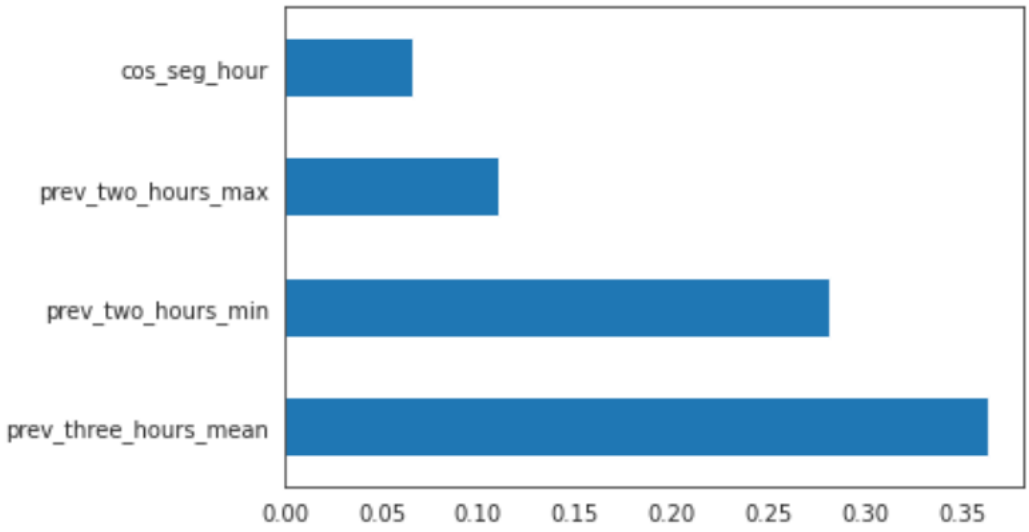
```
sns.set_style("white")
shap.summary_plot(shap_values)
```



plot feature importance

```
x=data[['Eastward-Moving air speed',
 'Northward-Moving air speed',
 'Temperature',
 'Surface solar radiation',
 'Total precipitation',
 'average_speed',
 'cos_seg_hour',
 'sin_seg_hour',
 'day',
 'month',
 'weekday',
 'weekend',
 'prev_two_hours_mean',
 'prev_two_hours_min',
 'prev_two_hours_max',
 'prev_three_hours_mean',
 'prev_three_hours_min',
 'prev_three_hours_max',
 'prev_2hours','day/night']]
```

```
(pd.Series(model.feature_importances_, index=x.columns)
   .nlargest(4)
   .plot(kind='barh'))
```

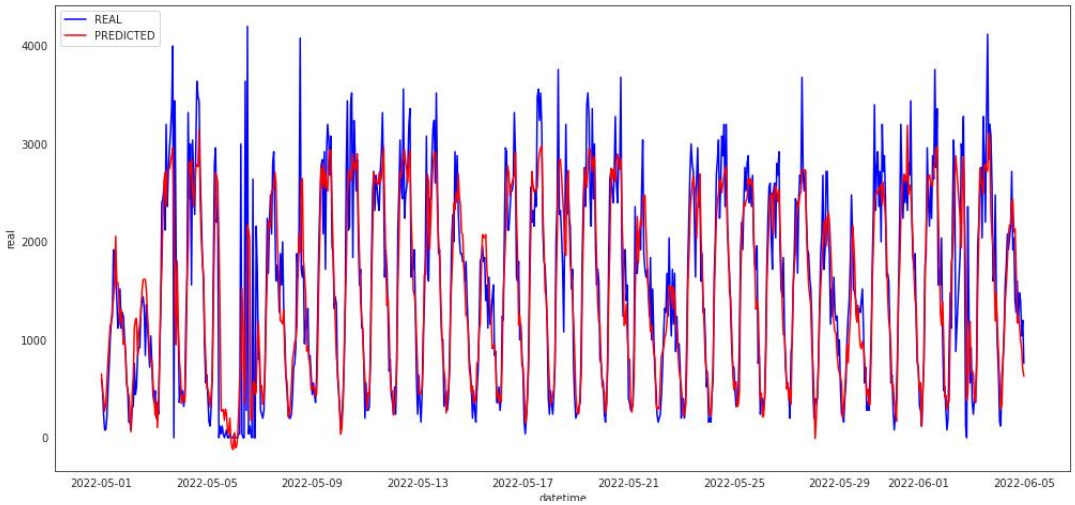


Predicted and actual data comparison plot

```
data_compare=pd.DataFrame({'real':y_test,'predicted':y_pred})
data_compare=data_compare.set_index(test.iloc[:,3])

fig,ax=plt.subplots(figsize=(17,8))

sns.lineplot(data=data_compare,x=data_compare.index,y=data_compare.rea
l,color='blue',label='REAL')
sns.lineplot(data=data_compare,x=data_compare.index,y=data_compare.pre
dicted,color='red',label='PREDICTED')
plt.legend()
plt.show()
fig.savefig('Real-Predicted')
```



12. References

[1]

Copernicus Information URL:

https://www.copernicus.eu/en/about-copernicus

[2]

Carlo Buontempo, Ronald Hutjes, Philip Beavis et. al 2019 ‘Fostering the development of climate change service (C3S) for agriculture applications

[3]

Climate Data Store (CDS) URL:

https://confluence.ecmwf.int/display/CKB/Climate+Data+Store+%28CDS%29+API+Keywords

[4]

Github netCDF4 data type URL:

https://unidata.github.io/netcdf4-python/

[5]

Yue Hou, Zhiyuan Deng, and Hanke Cui 2021 ‘Short-term traffic flow prediction with weather conditions: Based on deep learning algorithms and data fusion’

[6]

Copernicus Atmosphere Monitoring URL:

https://insitu.copernicus.eu/state-of-play/copernicus-atmosphere-monitoring-service

[7]

Copernicus Satellites URL:

https://www.copernicus.eu/en/about-copernicus/infrastructure/discover-our-satellites

[8]

Climate Data Store Documentation URL:

https://cds.climate.copernicus.eu/toolbox/doc/index.html

[9]

The Copernicus Marine Service URL:

https://www.copernicus.eu/en/copernicus-services/marine

[10]

The Copernicus Land Service URL:

https://www.copernicus.eu/en/copernicus-services/land

[11]
The Copernicus Security Service URL:
https://www.copernicus.eu/en/copernicus-services/security
[12]
Grib Data Type URL:
https://weather.gc.ca/grib/what_is_GRIB_e.html
[13]
ERA5-Land hourly data from 1950 to present Dataset URL:
https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-land?tab=overview
[14]
Impact Of Copernicus URL:
https://www.copernicus.eu/en/about-copernicus/impact-copernicus
[15]
Irrigation Management Case Study URL:
https://www.copernicus.eu/sites/default/files/UseCase_Agriculture_Irrigation.pdf
[16]
Energy Case Study URL:
https://www.copernicus.eu/sites/default/files/UseCase_Energy_OilAndGas.pdf
[17]
Solar Power Production Case Study URL:
https://www.copernicus.eu/sites/default/files/UseCase_Energy_SolarPowerProduction.pdf
[18]
Health Case Study URL:
https://www.copernicus.eu/sites/default/files/UseCase_Health_AirQualityMonitoring.pdf
[19]
Transportation Case Study URL:
https://www.copernicus.eu/en/about-copernicus/impact-copernicus/transport
[20]
Urban Planning Case Study URL:

https://www.copernicus.eu/sites/default/files/UseCase_UrbanPlanning_ConstructionMonitoring.pdf
[21]
Agricultural Case Study URL:
https://www.copernicus.eu/sites/default/files/UseCase_Insurance_AgriculturalInsurance.pdf
[22]
Athanasios I. Salamanis, Anastasia-Dimitra Lipitakis, George A. Gravvanis et. al 2021 'An adaptive cluster-based spare autoregressive model for large-scale multi-step traffic forecasting'
[23]
Xuexin Bao, Dan Jiang, Xuefeng Yang, Hongmei Wang. 2020 'An improved deep belief network for traffic prediction considering weather factors'
[24]
Federal Highway Administration
https://www.fhwa.dot.gov/policyinformation/statistics.cfm
[25]
Benjamin Lartey, Abenezer Girma, Abdollah Homaifair et. al 2021
'XGBoost: a tree-based approach for traffic volume prediction
[26]
Y. Wang, Y. Guo 2020 'Forecasting method of stock market volatility in time series data based on mixed model of arima and xgboost'
[27]
W. Wang, Y. Shi, G. Lyu, W, Deng 2017 'Electricity consumption prediction using xgboost based on discrete wavelet transform'
[28]
Misganaw Abebe, Yoojeong Noh, Young-Jin Kang et. al 2021
'Ship tranjectory planning for collision avoidance using hybrid ARIMA – LSTM models'
[29]
Lei xiong, Ye Yao 2021
'Study on an adaptive thermal comfort model with K-nearest-neighbors (KNN) algorithm.'
[30]

V. Vapnik, O. Chapelle 2000
'Bounds on Error Expectation for Support Vector Machines'.
[31]
Shuang Li, Kun Xu, Guangzhe Xue et al. 2022
'Prediction of coal spontaneous combustion temperature based on improved grey wolf optimizer algorithm and support vector regression'.
[32]
Pallabi Saikia, Rashmi Dutta Baruah, Sanjay Kumar Signh, Pradip Kumar Chadhurri 2020
'Artificial Neural Networks in the domain of reservoir characterization: A review from shallow to deep models'
[33]
Taghreed Alghamdi, Khalid Elgazzar, Magdi Bayoumi et al. 2019
'Forecasting Traffic Congestion Using Arima Modeling'
[34]
Themistoklis Diamantopoulos, Dionysios Kehagias, Felix G. Konig et al. 2013
'Investigating the effect of global metrics in travel time forecasting'
[35]
Athanasios Salamanis, Dionysios Kehagias, Christos D. Kehagias, et al. 2016
'Managing Spatial Graph Dependencies in Large Volumes of Traffic Data for Travel-Time Prediction'
[36]
Peibo Duan, Guoquiang Mao, Weifa Liang, Degan Zhang et al. 2019
'A Unified Spatio-Temporal Model for Short-Term Traffic Flow Prediction'
[37]
Jianhua Guo, Wei Huang, Billy M. Williams 2014
'Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification'
[38]
Afhshin Abadi, Tooraj Rajabioum, Petros A. Ioannou 2015
'Traffic Flow Prediction for Road Transportation Networks with Limited Traffic Data'
[39]
Kamarianakis, Wei Shen, Laura Wynter 2012

‘Real-time road traffic forecasting using regime-switching space-time models and adaptive Lasso’
[40]
Jia Zheng Zhu, Jin Xin Cao, Yuan Zhu 2014
‘Traffic volume forecasting based on radia basis function neural network with the consideration of traffic flows at the adjacent intersections’
[41]
Pinlong Cai, Yunpeng Wang, Guangquan Lu et al 2016
‘A spatiotemporal correlative k-nearest neighbour model for short-term traffic multistep forecasting’
[42]
Bin Sun, Wei Cheng, Prashant Goswami, Guohua Bai 2017
‘Short-term forecasting using self-adjusting k-nearest neighbours’
[43]
Fangce Guo, John W. Polak, Rajesh Krishman 2018
‘Predictor fusion for short-term traffic forecasting’
[44]
Wenbin Hu, Liping Yan, Kaizeng Liu et al 2016
‘A short-term Traffic Flow Forecasting Method Based on the Hybrid PSO-SVR’
[45]
Yuliang Cong, Jianwei Wang, Xiaolei Li 2016
‘Traffic Flow Forecasting by a Least Squares Support Vector Machine with a Fruit Fly Optimization Algorithm’
[46]
Baozhen Yao, Chao Chen, Quingda Cao 2016
‘Short-Term Traffic Speed Prediction for an Urban Corridor’
[47]
Jia Zheng Zhu, Jin Xin Cao, Yuan Zhu 2014
‘Traffic volume forecasting based on radia basis function neural network with the consideration of traffic flows at the adjacent intersections’
[48]
Jiaqiu Wang, Ioannis Tsapatakis, Chen Zhong 2016
‘A space-time delays neural network model for travel prediction’

[49]
Gaetano Fusco, Chiara Colombaroni, Natalia Isaenko 2016
'Short-term speed predictions exploiting big data on large urban road networks'
[50]
Jinjun Tang, Fang Liu, Yajie Zou, et al 2017
'An improved fuzzy neural network for traffic speed prediction considering periodic characteristics'
[51]
A. Ladino, A.Y Kibangou, C. Canudas de Wit, et al 2017
'A real time forecasting tool for dynamic travel time from clustered time series'
[52]
Kui-Lin Li, Chun-Jie Zhai, Jian-Min Xu 2017
'Short-term traffic flow prediction using a methodology based on ARIMA and RBFANN'
[53]
Jianhua Guo, Zhao Liu, Wei Huang, et al 2017
'Short-term traffic flow prediction using fuzzy information granulation approach under different time intervals'
[54]
Raza A., and Zhong M. (2018)
'Hybrid artificial neural network and locally weighted regression models for lane-based short-term urban traffic flow forecasting'
[55]
Mohammed Amine Naji, Sanaa El Filali, Kawtar Aarika, et al 2021
'Machine learning algorithms for breast cancer prediction and diagnosis'
[56]
Jitendra Singh Kushwah, Atul Kumar, Subhash Patel 2021
'Comparative study of regressor and classifier with decision tree using modern tools'
[57]
Ruoran Wang, Luping Wang, Jing Zhang, 2022
'Xgboos machine learning algorithms performed better than regression models in predicting mortality of moderate to severe traumatic brain injury'
[58]

P. Kulkarni, S. Londhe, M. Deo, 2017
‘Artificial Neural Networks for Construction Management’
[59]
Yasunobu Nohara, Koutarou Matsumoto, Hidehisa Soejima et al 2022
‘Explanations of machine learning models using shapley additive explanation and application for real data in hospital’
[60]
Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. Wiley Series in Probability and Statistics.
[61]
Goodfellow, I., Bengio, Y. and Courville, A., 2016. Machine learning basics. Deep learning, 1(7), pp.98-164.
[62]
Saravanan, R. and Sujatha, P., 2018, June. A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 945-949). IEEE.
[63]
Wang, D., 2001. Unsupervised learning: foundations of neural computation. Ai Magazine, 22(2), pp.101-101.