EMS 702P

Queen Mary University of London

CW - 2

TAXI TIME PREDICTION

by Machine Learning

Statistical Thinking and Machine Learning

Group B:

Christen Spencer

Abdullah Nasir

Kaiyuan Chen

Yufan Xia

TABLE OF CONTENTS

Introduction

Description

Accurate taxi time prediction has been critical in optimising airport airside operations. It is not only useful for practitioners to develop enhanced schedules and detect choke points between gate and runway, but it also assists government analysts in estimating appropriate airport capacity and evaluating the regulatory consequences.

The subject matter makes use of taxiing data from Manchester International Airport (MAN), which is the second busiest airport in the United Kingdom. The data in this case study contains up to 25 characteristics, with the goal of providing a suitable amount of information for taxi time prediction.

Individual Contribution

Yufan Xia – Neural Network (NN) Model

He was responsible for implementing, training, and validating neural network models. He also applied this supervised learning to predict the taxi time using the back propagation and least squares estimation algorithms along with analysing the performance of the NN model and prepare a separate result report.

Christen Spencer – Liner Regression (LR) Model

He worked on the implementation, training, and validation of linear regression models. Additionally, he utilized supervised learning techniques, employing both backpropagation and least squares estimation algorithms, to forecast taxi times followed by analysing the performance of the LR model.

Kaiyuan Chen – Adaptive Neuro Fuzzy Inference Systems (ANFIS) Model

He worked on the implementing, training, and validating ANFIS models. Applying this supervised learning to predict the taxi time using the back propagation and least squares estimation algorithms. He listed the advantages, disadvantages, accuracy, generalization ability, model transparency of the model and prepare a separate result report.

Abdullah Nasir – Comprehensive Analysis, Conclusion, and Report Preparation

He was responsible for summarizing and comparing the performance of the three models. He also integrated the work of all team members along with writing the final deliverable report, including introduction, dataset description, model description, comparative analysis, and conclusion.

Benefits of Predictive Model

Implementing a predictive model for airplane taxi time can offer numerous benefits to Air Traffic Management (ATM), leading to improved efficiency and operational effectiveness. Here's an overview of the advantages and the transformative impact it can have on ATM, along with future prospects.

Benefits:

I. Operational Efficiency:

Predictive models enable better estimation of taxi times, helping to minimize delays on the ground. Airlines and airports can optimize the allocation of resources, including ground staff, fuelling, and gate availability.

II. Cost Savings:

Airlines can optimize engine start times and taxi routes, leading to fuel savings. Efficient taxi time predictions contribute to better utilization of airport resources, potentially reducing operational costs.

III. Enhanced Passenger Experience:

Passengers experience shorter wait times on the tarmac, contributing to an improved overall travel experience. Predictable taxi times contribute to more accurate arrival time estimations for passengers.

IV. Integration with Traffic Management Systems:

Integration enables a more holistic approach to traffic management, considering both air and ground movements. The analysis of historical and real-time data helps monitor the performance of the entire aviation system, identifying areas for improvement.

Future Prospects:

I. Integration with Autonomous Systems:

As autonomous systems become more prevalent, predictive models can play a crucial role in optimizing the movements of autonomous ground vehicles and aircraft.

II. Collaborative Decision-Making:

Prospects involve developing collaborative decision-making frameworks, where predictive models are part of a networked system that includes airlines, airports, and air traffic control.

III. Environmental Considerations:

Predictive models can be used to optimize taxi routes, reducing fuel consumption, and contributing to the overall sustainability of air travel.

In conclusion, a predictive model for airplane taxi time has the potential to revolutionize Air Traffic Management by enhancing operational efficiency, reducing costs, and improving the overall travel experience. Continuous advancements in technology and data analytics will likely play a key role in shaping the future of predictive modelling in aviation.

Data Set

We utilized 1000 data sets from June and July and used two outlier detection and processing algorithms to clean the data. We took following path for processing the available data.

Data Selection

The data selection process is the process of selecting certain features from a feature set and inputting them into a PCA program.

Principle: Retain the features most relevant to aircraft taxiing time, and the correlation between these features must be low. Although taxi time seems a promising feature, but it is the target variable and should not be selected as a feature.

Process: Exclude obvious unrelated features, exclude features with correlation coefficients greater than 0.9 in correlation analysis, and remove features with small contributions in the principal components. Finally, the following seven features were retained.

Pre-processing for PCA

Data standardization

Standardization is a key step in data preprocessing, especially before conducting PCA. This is because PCA is a technique based on the covariance matrix of data, which aims to identify the main direction of change in the data. If the features in the dataset have different dimensions and numerical ranges, then features with larger dimensions may have a disproportionate impact on PCA results.

Calculating the Principal Components

The program calculates the principal components of the data using the PCA() function.

```
# Create PCA objects and perform PCA fitting on the data
pca_full = PCA()
X_pca_full = pca_full.fit_transform(X_scaled)
```

Naming the Principal Components

Determine the number of principal components to possess at least 80% of the variance criteria via analysis of “cumulative_explanatory_variance”

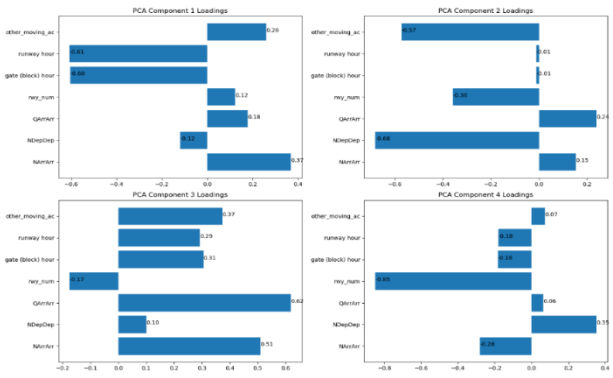
```
# Compute cumulative explained variance
cumulative_explained_variance_full = np.cumsum(explained_variance_ratio_full)

# Find the number of principal components that explain more than 80% variance
n_components_80 = np.argmax(cumulative_explained_variance_full >= 0.8) + 1

# Print the number of components to explain at least 80% variance
print(f"Number of components to explain 80% variance: {n_components_80}")
```

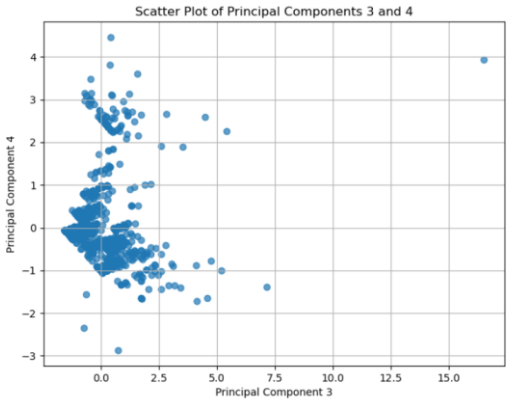
Interpreting the principal components

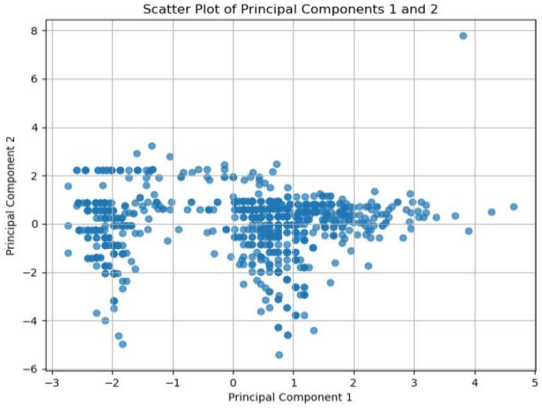
The following program prints the explained variance ratio for each principal component and visualizes the feature weights for each component in bar charts.



Plotting selected principal components

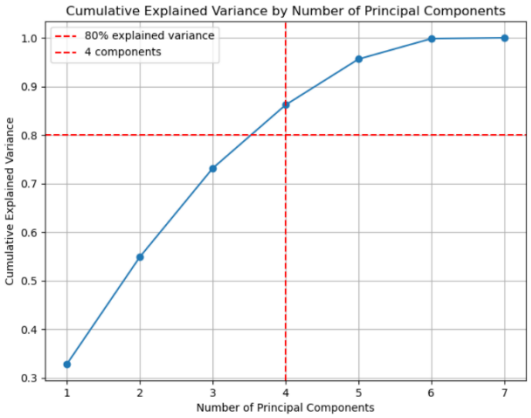
We generated graphs, to show how much of the total variance is retained by including more principal components as well as two scatter plots of principal components pairs.





Assessing the information loss

Information loss is indirectly assessed by printing the explained variance ratios. While the program doesn't provide specific values for information loss, it allows inference based on the explained variance ratios and cumulative explained variance curve. Based on the weights of the PCAs and the cumulative explained variance, it can be inferred that 13% of information is lost.



Models

Neural Network

Neural network is a powerful machine learning model, which is one of the core technologies in the current field of artificial intelligence and machine learning, and especially excels in dealing with complex pattern recognition and prediction problems.

In this task, the processed data needs to be passed into the input layer of the neural network, and then based on the processing of the hidden layer, the predicted value of taxi_time is output, after adjusting the learning rate, the number of training rounds, the activation function, the optimiser, and other hyperparameters to reduce the prediction error of the model.

The project uses PyTorch framework to build the neural network model, which is integrated with Python and easy to use. In the network layers, three fully connected layers are used, a fully connected layer is the most basic type of layer in a neural network, which multiplies the output of the previous layer with the weights of each neuron in the current layer and adds a bias term, and then an activation function introduces nonlinearity.

The mathematical operations of such a layer can be expressed as：

$$output=activation(W \cdot input+b)$$

The first fully connected layer maps the input features to 32 dimensions, the second to 16 dimensions and the last to the output feature dimensions. The SiLU activation function is used between the fully connected layers, which helps to introduce non-linearity, allows the network to capture more complex data patterns and helps to optimise the flow of gradients during the process, and has the mathematical form: SiLU(x)=x·σ(x)

Where：

$$\sigma(x)=\frac{1}{1+e^{-x}}$$

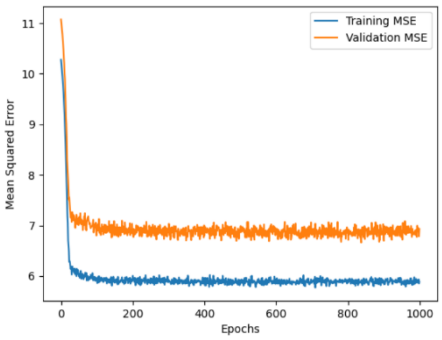
In the fully connected layer to address the overfitting problem Dropout is introduced to reduce overfitting. Overfitting is a common problem in machine learning, where a model learns too much detail and noise on the training data, which causes the model to perform poorly on new, unseen data. Each time an input passes through the network, it is actually computed on a slightly different network due to the Dropout, which can be approximated as the effect of training multiple networks and integrating them.

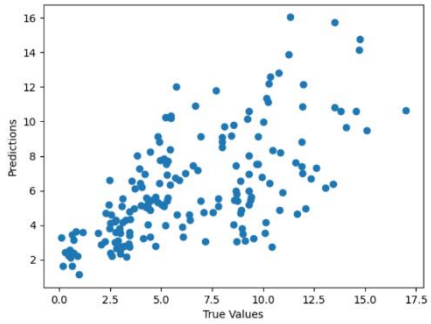
In the backpropagation algorithm in this project, Smooth L1 Loss from the PyTorch library is used as the loss function, which is suitable for data with a high number of outliers, and is stable when the gradient is small, which facilitates stable training. Adamax is used as the optimiser, which is based on the Adam algorithm for infinite paradigms. The combination of Smooth L1 Loss and Adamax is designed to find a balance between stability and robustness to outliers, while taking advantage of the potential benefits of Adamax on sparse data. After then calculating the gradient of the loss function with respect to each parameter, Adamax uses the gradient to automatically update the model parameters and reduce the overall loss value.

0.8 of the dataset is used as the training set and the rest as the test set. The number of training rounds i.e. epochs is then specified and at each epoch a series of iterations are performed on the training set. At the end of each epochs, iterations are performed on the validation set to calculate and validate the loss and the metrics are used as MSE (Mean Square Error). Eventually, at the output, the current training rounds and MSE metrics are printed every 10 epochs.

Results

After a series of tuning parameters can be found out the best performance parameters of the model, the following are the performance results of the neural network model on the training set and validation set of this project. It should be noted that the performance of the training set is better than performance of the test set, which indicates that there is a little bit of overfitting. At the same time, the two MSE values eventually stabilise at 7 and 6 respectively, which means that the model predictions are not perfect and there is an acceptable error, which is also very much dependent on the quality and quantity of the dataset.





The predicted values are compared to the real data values. Ideally, the data points should be clustered into a 45° straight line, but as the figure shows, the model's error is not small enough, resulting in a more pronounced deviation of the predicted values from the real data.

Linear Regression

Introduction

This study employs linear regression, a statistical methodology for predicting a continuous variable based on independent variables, with a specific application to forecasting taxi time using Principal Component Analysis (PCA) features. The model assumes a linear relationship, expressed as y = mx + b, where y is the dependent variable, x is the independent variable, m is the slope, and b is the y-intercept. The optimization objective is to determine values for m and b that minimize the disparity between predicted and actual values, typically gauged through metrics such as Mean Absolute Error (MAE).

The current investigation assesses the efficacy of a linear regression model for predicting taxi time, emphasizing the incorporation of PCA features. Initial observations indicate suboptimal performance marked by elevated MSE, attributable to deficiencies in dataset mining and manipulation. The dataset's inherent limitations compromise its suitability for insightful data mining, adversely affecting the linear regression model.

Methodology

To rectify these issues, efforts are underway to enhance the model's performance by introducing novel features. Emphasis is placed on the inclusion of a variable indicating airport congestion during both take-offs and landings. This augmentation is anticipated to bolster the model's capacity to derive meaningful conclusions.

Dataset Truncation

The dataset undergoes truncation to 950 data points for PCA fitting, with the exclusion of the last 50 data points representing July's information. This truncation introduces a potential source of inaccuracy, exacerbating existing flaws resulting from incorrectly mined features.

Result

In conclusion, the linear regression model's suboptimal performance stems from inadequacies in dataset mining. To enhance reliability, this study proposes the introduction of new features, emphasizing the assessment and augmentation of data quality. It is acknowledged that current limitations in chosen features and data manipulation techniques necessitate further refinement in future iterations of the model.

ANFIS

Introduction

ANFIS is based on a fuzzy inference system, which first derives a set of fuzzy rules and subsequently transforms inputs into outputs. It then constructs a neural network in a layered manner, with each layer corresponding to a step in fuzzy inference, including fuzzification, rule application, normalization, defuzzification, and output calculation. One of the methods for constructing fuzzy rules is through clustering, where data is grouped, and each group represents a rule.

In summary, ANFIS combines the principles of fuzzy logic with neural networks, and it adapts its rule parameters based on input-output data, with clustering being one of the methods used to construct fuzzy rules tailored to specific requirements.

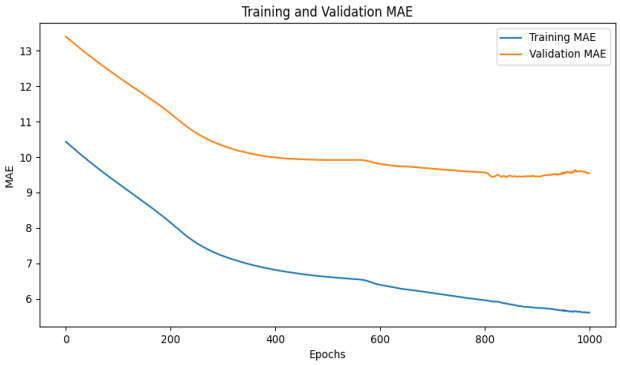
Methodology

The ANFIS code you mentioned consists of a fuzzy layer for rule generation, a rule layer for rule application, and a linear layer for generating the output. Regarding the dataset, the code splits it into two parts, one for testing and the other for training, typically in a 20-80 ratio (20% for testing and 80% for training). Additionally, the code uses Mean Absolute Error (MAE) as the metric to present the results, and it associates MAE with the number of training epochs to show how MAE changes with training progress.

As a classification method, ANFIS provides a certain level of transparency through its use of fuzzy rules, making it more interpretable compared to a pure neural network. However, the integration of neural networks adds complexity, which can reduce the overall model's transparency. These code practices, such as calculating MAE, demonstrate the model's ability to handle data outside the training dataset effectively.

Result

The result of MAE after 1000 epochs we can see the Validation MAE is always bigger than Training MAE though they are both decreasing. This means after each training the accuracy is increasing each time also the model is fitting in the training data and can generalizing to new data.



Due to the insufficient capacity the project may have a problem of under-fitting some features have been changed during the testing trying to solve this problem however they did not have some huge effect, but the result has shown the trend of the training at least.

Conclusion

In the realm of predicting taxi time at airports using Python, the choice between neural networks, linear regression, and ANFIS involves a nuanced consideration of each model's characteristics. Neural networks, with their intricate architecture, excel at capturing complex, nonlinear relationships within the data. However, this flexibility comes at the cost of potentially requiring a substantial dataset for effective training, and their interpretability can be challenging. On the other hand, linear regression, a simple and interpretable model assuming a linear relationship, proves efficient and transparent, making it suitable when the underlying patterns are relatively straightforward. However, it may struggle to capture the complexities of nonlinear patterns present in taxi time data. ANFIS, a hybrid model blending neural networks with fuzzy logic, stands out for its adaptability to handle uncertainties in the data, making it effective when dealing with both numerical and symbolic information. The interpretability of ANFIS falls between the transparency of linear regression and the complexity of neural networks, making it a valuable compromise. Ultimately, the choice among these

models hinges on factors such as the dataset's size, the interpretability required, and the nature of relationships within the data, with each model offering unique strengths and trade-offs in the pursuit of accurate taxi time predictions.

Due to feature engineering limitations has caused a hindrance towards reducing error. The break off can be avoided with providing more features for specifics.

References

1. www.eurocontrol.int. (n.d.). *FLY AI Report*. [online] Available at: https://www.eurocontrol.int/publication/fly-ai-report
2. qmplus.qmul.ac.uk. (n.d.). *Log in to the site | MyQMUL*. [online] Available at: https://qmplus.qmul.ac.uk/pluginfile.php/3460516/mod_resource/content/1/AFE-87-Final-Report.pdf
3. Zou, J., Han, Y. and So, S.-S. (2008). Overview of Artificial Neural Networks. *Methods in Molecular Biology™*, 458, pp.14–22. doi: https://doi.org/10.1007/978-1-60327-101-1_2
4. Ravizza, S., Chen, J., Atkin, J.A.D., Stewart, P. and Burke, E.K. (2014). Aircraft taxi time prediction: Comparisons and insights. *Applied Soft Computing*, 14, pp.397–406. doi: https://doi.org/10.1016/j.asoc.2013.10.004