Team ID: 1 - SC

كيرلس نبيل منير فهمي 20191700460 SC

خالد احمد عبد الظاهر 20191700221 SC

ندي مجدي عبد الكريم 20191700685 SC

يوسف نادر ميشيل صبحي 20191700793 SC

ابانوب جمال فكري فوزي 20191700001 SC

AIRLINE TICKET PRICE PREDICTION (REGRESSION)

2021 2022

# Introduction

The project aims to predict the price of the flight ticket with the least error. This approach will be achieved by using two regression techniques

* Multiple Regression
* Polynomial Regression

# Pre-Processing

Pre-Processing is essential step in any ML process for success of the model.

Our processing is divided into several steps:

* X\_preprocessData ()
* Y\_preprocessData ()
* Feature encoding
* Feature scaling

## X\_preprocessData ()

This function processes the feature to apply the regression techniques on it. The number of features in the " airline-price-prediction.csv" is "10" features

{"date", "airline", "ch\_code", "num\_code", "dep\_time", "time\_taken", "stop", "arr\_time", "type", "route"}

### Date Column

* The data format "x/x/2022" and "x-x-2022"
* We iterate in all data and check if data has "-" and "/" then split using "strptime ()" function that return datetime object (day, month, year)
* Append day in day list and month in month list and ignore year because all rows have the same value in dataset "2022" to avoid data redundant
* Drop column date and make columns for day and month and assign its values from day list and month list.
* Convert 2 Columns from string to numeric.

### Dep\_Time Column

The values of Dep\_Time is in form xx: xx, so the technique is

* Replace ":" to "." To act as float number.
* Convert Column from string to numeric using "to\_numeric(X[dep\_time])".
* Finally covert data from “xx: xx” to xx.xx numeric.

### Time Taken Column

By reviewing our data, we found that all rows in format "08h 05m" but there with one mis-format "1.01h m" in row 91882 founded by "X['time\_taken'].value\_counts ()" function to found unique values and solved by:

* Remove “h m” -> Special Case founded in one only row.
* Replace "h "to "." To act as float number.
* Remove “m” to make sure the data become in format “xx.xx” only.
* Convert Column from string to numeric using "to\_numeric(X[time\_taken]) ".

### Stop Column

* The value of Stop Column is one of three (1-stop or 2+-stop or non-stop), so the technique is:
* Split values by "– " and select index 0 in spilt.
* Replace "non" by "0" and “2+” by “2”.
* Convert Column from string to numeric using "to\_numeric(X['stop']) ".
* Finally covert data from “xx: xx” to xx.xx numeric.

### Arr\_Time Column

The values of Dep\_Time is in form xx: xx, so technique is:

* Replace “:” to “.” To act as float number
* Convert Column from string to numeric using “to\_numeric(X[arr\_time])”
* Finally covert data from “xx: xx” to xx.xx numeric

### Route Column:

values such as {'source': 'Delhi', 'destination': 'Hyderabad'}, technique is

* Split each row in two half by “,” into new Variable, index [0] = {'source': 'Delhi' , index[1] = 'destination': 'Hyderabad'}
* Foreach row in new, Split index [0] by “: “and append second part in source list, ex: ‘Delhi’ and Split index [1] by “: “, get 'Hyderabad'} to remove ‘}’ made another split by ‘}’ and append first part of second split to destination list
* Drop column route and make columns for source and destination and assign its values from source list and destination list.

### ('airline', 'ch\_code', 'type', 'source', 'destination') Columns

All Columns has strings values, so we use encoding Label to convert it from strings to numeric values

## Y\_preprocessData ()

This function is responsible for processing the "price" column (Y-label) the format of the column is string consists of sum digits with a comma ','

The processing aims to change the format of this string to make it a number so it can be used in the model and fitting.

Text

Description automatically generated

Figure :Y\_preprocessData () function

## Feature encoding

The idea of feature encoding is the inability of applying the mathematical operations on the categorical columns such as (airline, ch\_code, type, source, and destination) which are include string values, so we encode these columns using “LabelEncoder” technique to convert it into numerical columns.

Text

Description automatically generated

Figure :Feature encoding function

## Feature scaling

The idea of feature scaling is to make the features on a similar scale.

Our approach is to use "Min-Max Normalization"

Min(X): the smallest value in the column.

Max(X): the largest value in the column.

Text

Description automatically generateda and b: are the range of numbers that the desired output wanted to be in.

Figure :Featur scaling function

# Feature analysis and selection

Feature analysis and selection is based on correlation in "Correlation.py" file as it calculates the correlation between all data after it processed producing a matrix with correlations with correlation greater than "0.1"

## Feature discarded

Our assumption is to remove any unrelated features with the price and remove any highly correlated features as it will be redundant and cause error increment

* Any Feature has a correlation less than 0.1 with price will be discarded
* Features "airline" and "ch\_code" are highly correlated as shown "0.88" so we choose only one of them "ch\_code" as it has higher correlation with "price" feature.

Feature "price" is doped from the "top\_feature" object to avoid redundant in data.

Our final features that will be used for model training are {'ch\_code', num\_code', 'time taken', 'stop', 'type'}

**Graphical user interface, application

Description automatically generated­­­**

Figure 4:Correlation heatmap

# Regression techniques:

Two regression techniques:

## Multi-variable regression

Hypothesis:

Our parameters are chosen as shown as in "Feature analysis and selection" section.

Using "sklearn" library to help us creating the model, our model consist of three main built-in functions

* LinearRegression ()
* fit ()
* predict ()

First create linear model using the " LinearRegression ()" function which returns a model that can be trained using "fit ()" function which is responsible for estimating the attributes out of the input data "x\_train, y \_train " and stores the model attributes and finally return the fitted estimator. "predict ()" will perform a prediction for each test "x\_test" instance

Text

Description automatically generated

Figure 5:multi-variable model output

## Polynomial regression

Hypothesis: =

Hyperparameters: degree of the polynomial (5)

This value achieves a reasonable and stable error.

Using "sklearn" library to help us creating the model, our model consist of three main built-in functions

* fit\_transform ()
* LinearRegression ()
* fit ()
* predict ()

Text

Description automatically generatedTransform the existing features to higher degree features using fit transform function this is the different and additional step from the multivariable regression.

Figure 6:Polynomial model output

## Time analysis

Time analysis is divided into two sections

* Training time
* Prediction time

Both are done using "Time ()" function calling this function before and after training and prediction and subtract the (start – end) time to get the actual time for both.

It is obvious that the time for training a multi-variable model is much less using a polynomial model due to the difference in complexity of polynomial techniques.

And as the degree of the polynomial increase the complexity increase so the time.

## Improvements

* Feature scaling
  + We tried different ranges of scaling it produces different accuracies but range from 0 --> 1 gives us fine accuracy
* Feature encoding
  + We tried different types of encoding like
    - One-hot encoding: this type makes every unique value in the column as a column itself and gives it value '1' if it exists in the row and '0' if not this method is rejected because it increases the data size, and the accuracy was in the same range.
    - Mean encoding: this type is rejected as the accuracy was in the same range of label Encoding but its implementation was harder, so it is not worth.
    - Label encoding: this is the used type in the project it gives us good accuracy and do not change the scale of the data.
* Drop nulls
  + Dropping nulls is only from training data to not affect the testing data number
  + To solve nulls in testing data we tried to approach
    - Replace nulls with zeros.
    - Replace nulls with the mean of the column.
* Degree of the polynomial
  + We tried to change the degree of the polynomial till the accuracy become stable degree = 5

# Data size

Our data are divided into two parts

* Training data
* Testing data

We divided the data into 80% for training and 20% for testing.

## Improvements

Text

Description automatically generated We tried different data sizes like 60% for training and 40% for testing.

Figure 7:Multi-variable model 40% testing

Text

Description automatically generated

Figure 8:.Polynomial model 40% testing

Text

Description automatically generated Higher accuracy when using 80% for training and 20% for testing.

Figure 9:Polynomial model 20% testing

Text

Description automatically generated

Figure 10:Multi-variable model 20% testing

# Conclusion

|  |  |  |
| --- | --- | --- |
|  | Multi-variable model | Polynomial model |
| MSE | 50961461.76891036 | 31726181.676215604 |
| R2\_Score | 0.9006508082342362 | 0.9381499196071196 |
| Training time | 0.03171110153198242 | 3.9175803661346436 |
| Testing time | 0.0014984607696533203 | 0.17902851104736328 |

## Model accuracy

Polynomial model has "MSE" and "R2 Score" less than multi-variable model so that polynomial model is better than multi-variable model.

Using highly correlated features in model training dec the accuracy of the model.

## Model time

Polynomial model has "Training time" greater than multi-variable model.