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

Milestone 2

Team ID: 1 - SC

AIRLINE TICKET PRICE CLASSIFICATION (SVM)

2021 2022

# Introduction

The project aims to classify the price of the flight ticket (very expensive – expensive – moderate - cheap ) with the least error. This approach will be achieved by using six regression techniques

* Linear SVM
* Polynomial SVM
* RBF SVM
* Linear kernel SVM
* Logistic Regression
* Decision Tree

# Pre-Processing

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

Our processing is divided into several steps:

* TicketCategory encoding ()

## TicketCategoryencoding

Text

Description automatically generatedThe idea of feature encoding is the inability of applying the mathematical operations on the categorical columns which are include string values, so we encode these columns by mapping each string value to a numeric value that expresses the weight of the real value.

Figure 1:Feature\_Encoder\_TicketCategory

## Feature selection

Correlation techniques in classification is the same as regression it differs only in the features there is only one additional feature "month".

Graphical user interface, chart

Description automatically generated

Figure :Correlation

# Classification techniques:

Six classification techniques:

## Linear SVM

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 3:Linear SVM model output

## Polynomial SVM

Hypothesis:

Hyperparameters: degree of the polynomial (3)

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 ()
* predict ()

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

Text

Description automatically generated

Figure 4:Polynomial model output

## RBF SVM

Hypothesis:

Text

Description automatically generated

Figure 5: rbf svm model output

## Linear kernel SVM

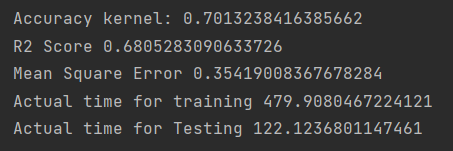


Figure : Linear kernel model

## Logistic Regression

Text

Description automatically generated

Figure 7: Logistic Regression

## Decision Tree

Text

Description automatically generated

Figure 8: Decision Tree

## 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 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 for only '**x**' features.
    - For Label 'Y' we mapped each string value to a numeric value that express it's weight.
      * Very expensive -> 3
      * Expensive -> 2
      * Moderate -> 1
      * Cheap -> 0
* Degree of the polynomial
  + We tried to change the degree of the polynomial till the accuracy become stable degree = 3
* Hyper parameter tuning
  + regularization parameter = 0.001
    - If c is large -> hard-margin classifier (best accuracy) "overfitting"
    - If c is small-> soft-margin classifier (more generalization)

# 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.

# Bar Graph

1. Train Time Graph For each Model
2. Test Time Graph For each Model
3. R2 Score Graph For each Model

# Conclusion

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Linear SVM model | Polynomial SVM model | RBF SVM model | Linear kernel model | Logistic Regression Model | Decision Tree Model |
| MSE | 0.322842512801298 | 0.287748220307231 | 0.29761458723616835 | 0.35419008367678284 | 0.28643686774072685 | 0.1831644798787983 |
| R2\_Score | 0.708803130879295 | 0.740457413287027 | 0.7315581666071992 | 0.6805283090633726 | 0.7416402245545998 | 0.8347896544731999 |
| Accuracy | 0.707755713750468 | 0.724116398151617 | 0.7240539527913076 | 0.7013238416385662 | 0.7183714250031222 | - |
| Training time | 0.965080976486206 | 459.9164102077484 | 881.3593971266022 | 479.9080467224121 | 1.8243277072906494 | 0.08701968193054199 |
| Testing time | 0.005949020385742 | 73.35036206245422 | 133.04427695274353 | 122.1236801147461 | 0.004501819610595703 | 0.0035033226013183594 |

One of the important conclusions is that not all high complex models always gives the best results it depends on many attributes like size of the data the relations between the data and also it depends on what we need sometimes we are concerned about time not the maximum efficiency.