2021
2022

TEAM ID: 1 - SC

AIRLINE TICKET PRICE CLASSIFICATION

MILESTONE 2

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

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

The 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.

```
def Feature_Encoder_TicketCategory(Y):
    l = []
    l = Y['TicketCategory']
    x = []
    for rows in range(len(l)):
        if l[rows] == 'very expensive':
            x.append(3)
        elif l[rows] == 'expensive':
            x.append(2)
        elif l[rows] == 'moderate':
            x.append(1)
        elif l[rows] == 'cheap':
            x.append(0)
    return x
```

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".

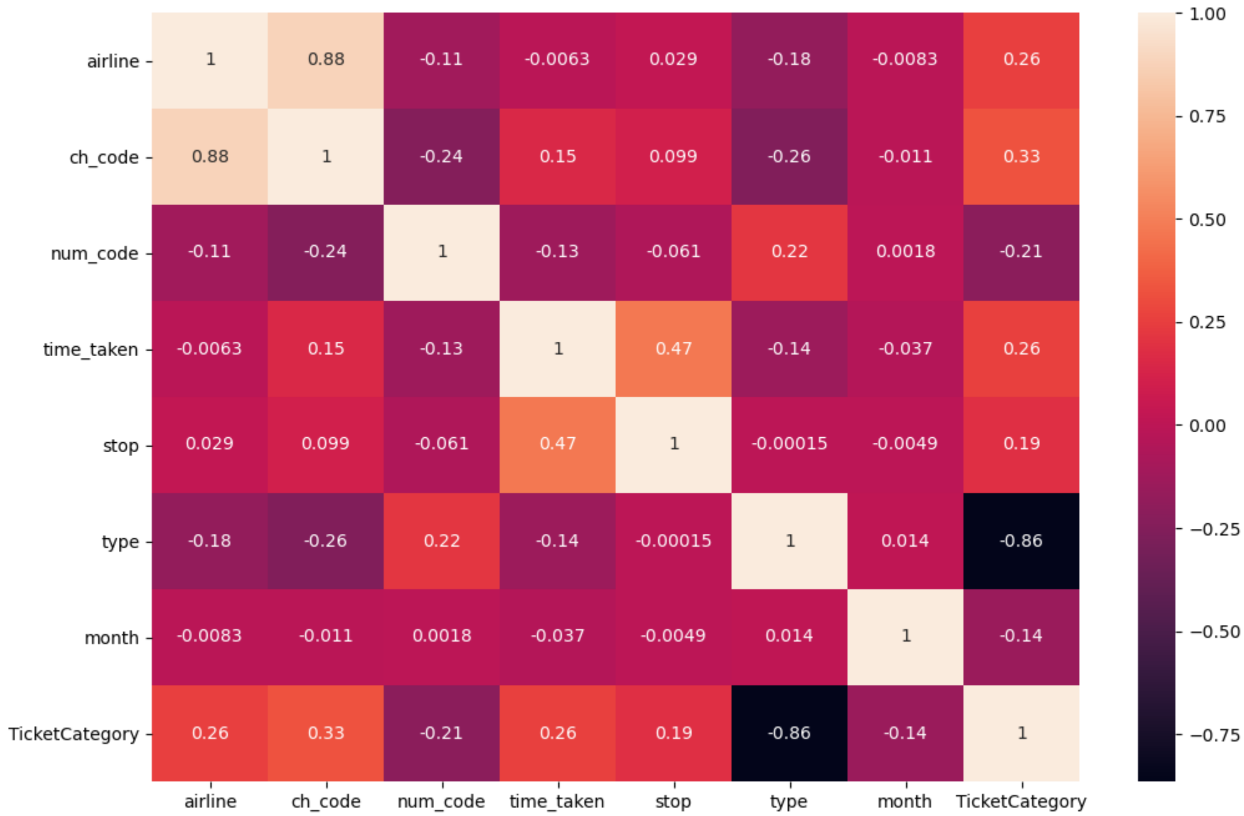


Figure 2:Correlation

Classification techniques:

Six classification techniques:

Linear SVM

Hypothesis: $h_\theta(x) = \min_\theta C\ \sum_i^m \left[y^i\ cost_1\ \left(\theta^T x^i\right) + \left(1 - y^i\right) cost_0\left(\theta^T x^i\right)\right] + \frac{1}{2} \sum_{j=1}^n\ \theta_j^2$

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

```
Accuracy linear: 0.7077557137504683
R2 Score 0.7088031307929542
Mean Square Error 0.3228425128012989
Actual time for training 0.965080976486206
Actual time for Testing 0.0059490203857421875
```

Figure 3:Linear SVM model output

Polynomial SVM

Hypothesis: $K\left(x_i, x_j\right) = \left(1 + x_i^T\ x_j\right)^2$

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.

```
Accuracy polynomial: 0.7241163981516173
R2 Score 0.7404574132870276
Mean Square Error 0.2877482203072312
Actual time for training 459.9164102077484
Actual time for Testing 73.35036206245422
```

Figure 4:Polynomial model output

RBF SVM

Hypothesis: $K(x_i, x_j) = \exp\,(-\frac{||x_i - x_j||^2}{2\sigma^2}$

```
Accuracy rbf: 0.7240539527913076
R2 Score 0.7315581666071992
Mean Square Error 0.29761458723616835
Actual time for training 881.3593981266022
Actual time for Testing 133.04427695274353
```

Figure 5: rbf svm model output

Linear kernel SVM

```
Accuracy kernel: 0.7013238416385662
R2 Score 0.6805283090633726
Mean Square Error 0.35419008367678284
Actual time for training 479.9080467224121
Actual time for Testing 122.1236801147461
```

Figure 6: Linear kernel model

Logistic Regression

```
Accuracy Logistic Regression: 0.7183714250031222
R2 Score 0.7416402245545998
Mean Square Error 0.28643686774072685
Actual time for training 1.8940999507904053
Actual time for Testing 0.005019187927246094
```

Figure 7: Logistic Regression

Decision Tree

```
R2 Score 0.8347896544731999
Mean Square Error 0.1831644798787983
Actual time for training 0.08701968193054199
Actual time for Testing 0.0035033226013183594
```

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
    - $C = \frac{1}{\lambda}$ 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

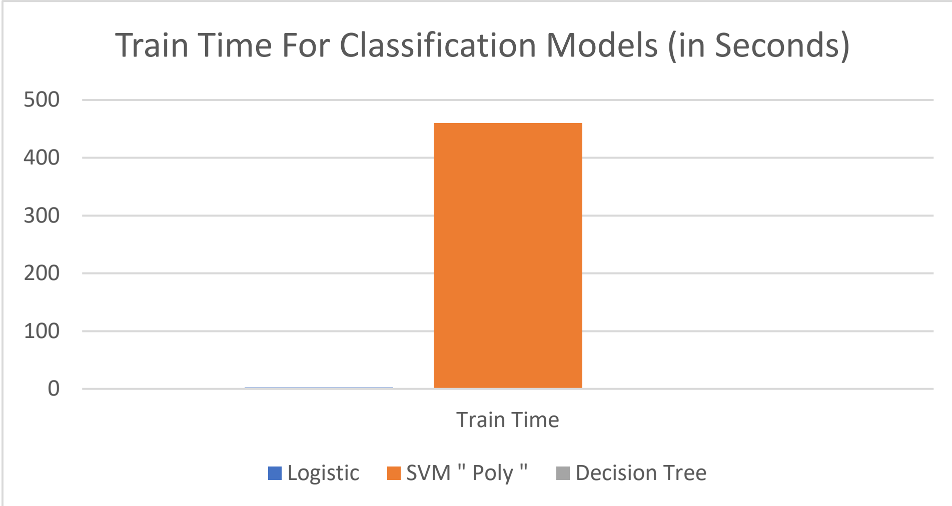
Our data are divided into two parts

- Training data
- Testing data

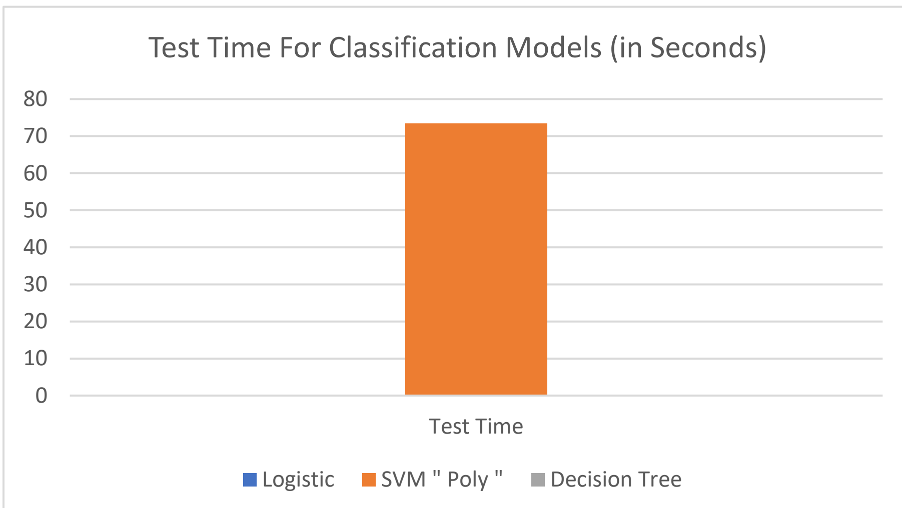
We divided the data into 80% for training and 20% for testing.

Bar Graph

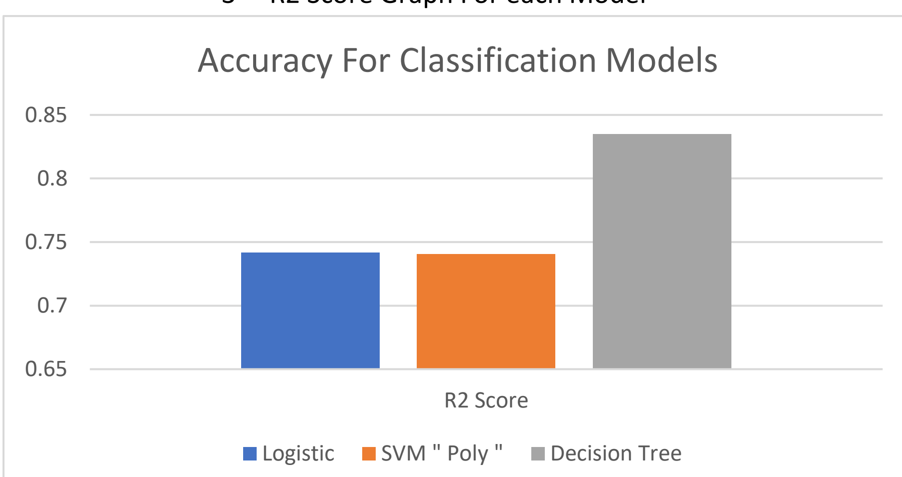
1- Train Time Graph For each Model



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 give 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.