<u>Pattern Recognition Milestone2</u>

CS_57 – Movies Popularity Prediction

Name	ID	Section	Department
مينا عادل لويز متى جرجس	20201700894	9	CS
مارلين تاوضروس يعقوب تاوضروس	20201701119	6	CS
مارتينا صبرى مسعد عجايبى	20201700626	6	CS
مينا نبيل اسعد نجيب	20201700897	9	CS
محمد تامر محمد محمدى	20201700677	7	CS

Preprocessing Techniques :-

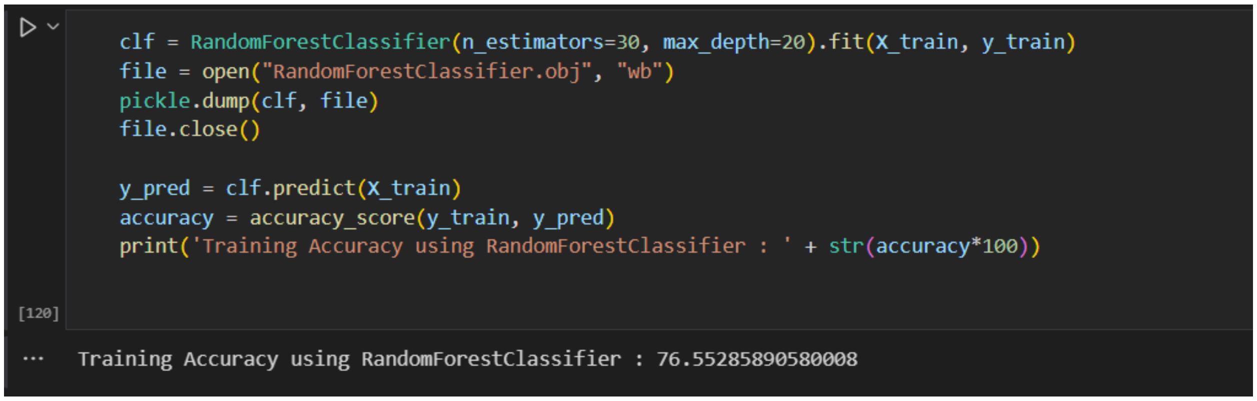
- We Applied **One hot Encoding** Technique on the following Columns **{Genres – Production Countries – Production Companies – Keywords – Spoken Language}** To determine the most effective values of these columns by separating each unique value into a new column and assign to it value = 1 in each row it appeared within otherwise the value = 0 so we can deal with categorical value in a numerical shape which make it more computable.
- We Applied **Feature Encoding** Technique on the following Columns **{Original Language – Original Title – Status}** To Convert Categorical Columns that have maximum of one value per record to numerical data so we can deal with it in a more computable way.
- We Applied **TF-idf Encoding** Technique on the following Columns **{Overview – Tagline}** to deal with them as they contain free text.

Some Analysis :-

- We Applied **Scaling** using **MinMaxScaler** on Numerical Columns
- We Applied **AnovaTest** On Numerical Columns and kept values less than 0.05
- We Applied **chi2 squared** On Categorical Columns and kept values less than 0.05

Classification Techniques :-

- **Random Forrest** : In our project, we employed the Random Forest Classifier algorithm to perform multi-label classification on a preprocessed text dataset. We trained the model using 30 estimators and a maximum depth of 20, and used the fit function to fit the model to the training data.
  We then used the trained model to predict the labels of the training data using the predict function, and calculated the accuracy of the model using the accuracy_score function. The resulting training accuracy achieved using the Random Forest Classifier was printed to the console, providing us with a quantitative measure of the performance of the model.



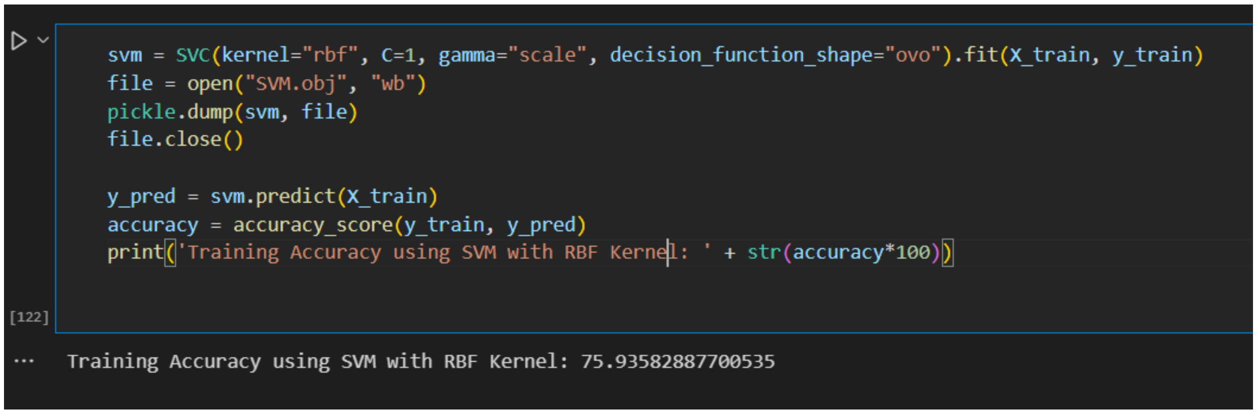
```
clf = RandomForestClassifier(n_estimators=30, max_depth=20).fit(X_train, y_train)
file = open("RandomForestClassifier.obj", "wb")
pickle.dump(clf, file)
file.close()

y_pred = clf.predict(X_train)
accuracy = accuracy_score(y_train, y_pred)
print('Training Accuracy using RandomForestClassifier : ' + str(accuracy*100))
```

[120]

```
Training Accuracy using RandomForestClassifier : 76.55285890580008
```

- **SVM** : In this SVM model, the hyperparameters are set as follows:
  kernel="rbf": This specifies that the SVM should use radial basis function (RBF) kernel for non-linear classification.
  C=1: This is the regularization parameter, which controls the trade-off between achieving a low training error and a low testing error. A smaller value of C will create a wider margin and may result in more errors but better generalization, while a larger value of C will create a narrower margin and may lead to overfitting. Here, C is set to 1, which is a moderate value.
  gamma="scale": This defines the gamma parameter of the RBF kernel. The gamma parameter defines how far the influence of a single training example reaches. A higher value of gamma leads to a tighter, more localized decision boundary, whereas a lower value of gamma leads to a smoother, softer decision boundary. Here, gamma is set to "scale", which means that it is calculated as 1 / (n_features * X.var()) by default.

decision_function_shape="ovo": This specifies that the one-vs-one (OvO) strategy should be used to handle multi-class classification.



```
svm = SVC(kernel="rbf", C=1, gamma="scale", decision_function_shape="ovo").fit(X_train, y_train)
file = open("SVM.obj", "wb")
pickle.dump(svm, file)
file.close()

y_pred = svm.predict(X_train)
accuracy = accuracy_score(y_train, y_pred)
print('Training Accuracy using SVM with RBF Kernel: ' + str(accuracy*100))
```

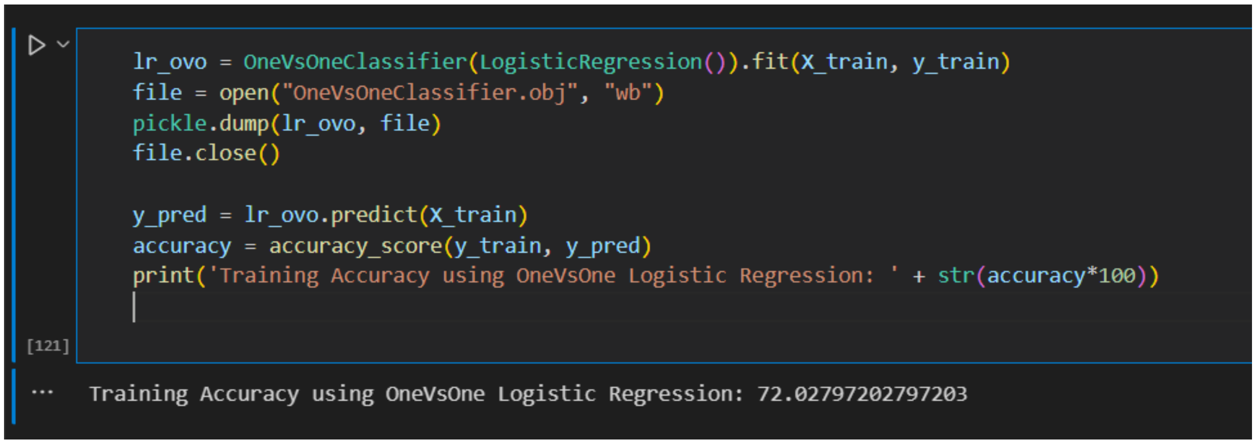
[122]

```
Training Accuracy using SVM with RBF Kernel: 75.93582887700535
```

Logistic Regression :-

we used the One-vs-One (OvO) technique in combination with Logistic Regression to perform multi-label classification on preprocessed text data. The OvO technique involves training a separate binary classifier for each pair of labels, and then combining the results of these classifiers to produce a multi-label classification.

To implement this technique, we used the `OneVsOneClassifier` function from the scikit-learn library to train a logistic regression classifier on the preprocessed training data.



```
lr_ovo = OneVsOneClassifier(LogisticRegression()).fit(X_train, y_train)
file = open("OneVsOneClassifier.obj", "wb")
pickle.dump(lr_ovo, file)
file.close()

y_pred = lr_ovo.predict(X_train)
accuracy = accuracy_score(y_train, y_pred)
print('Training Accuracy using OneVsOne Logistic Regression: ' + str(accuracy*100))
```

[121]

```
Training Accuracy using OneVsOne Logistic Regression: 72.02797202797203
```

Comparison between Models :-

Result / Model	**Random Forrest**	**SVM “RBF”**	**Logistic Regression**
Train Accuracy	76.55285890580008	75.93582887700535	72.02797202797203
Test Accuracy	72.5328947368421	71.38157894736842	67.92763157894737

Discarded Features :-

- **Home Page :** discarded because it contains many NULL Values (~=50% or more of the data).
- **ID :** discarded because it is irrelevant column.
- **Title :** discarded because it is a duplicate of the Original Title.
- **Any other feature selection was handled in the analysis section.**

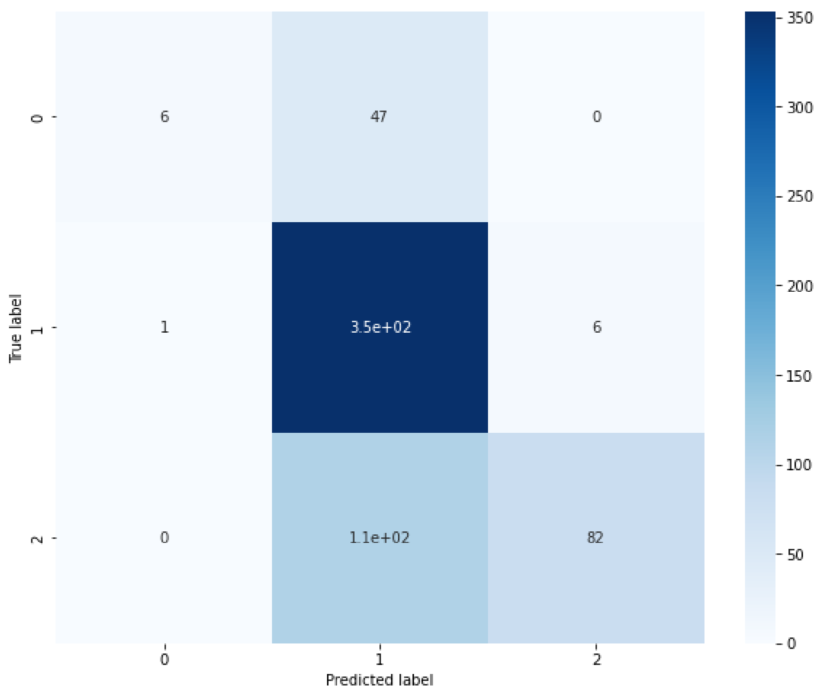
Important sizes : **Train size = 80% , Test size = 20% , Validation = 0%.**

Other Improvements :-

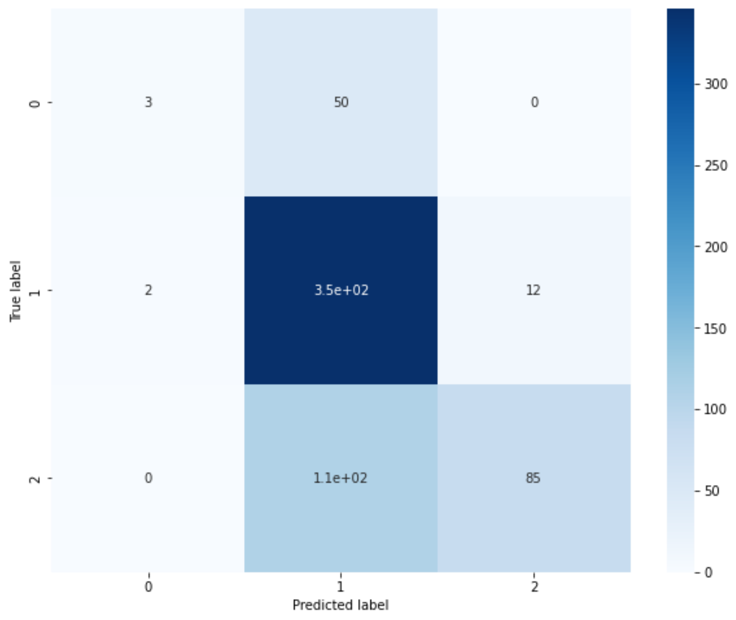
- We dropped the resulting columns from **one hot encoding** that had a sum of 1 or 2 .

Some Important Plots :-

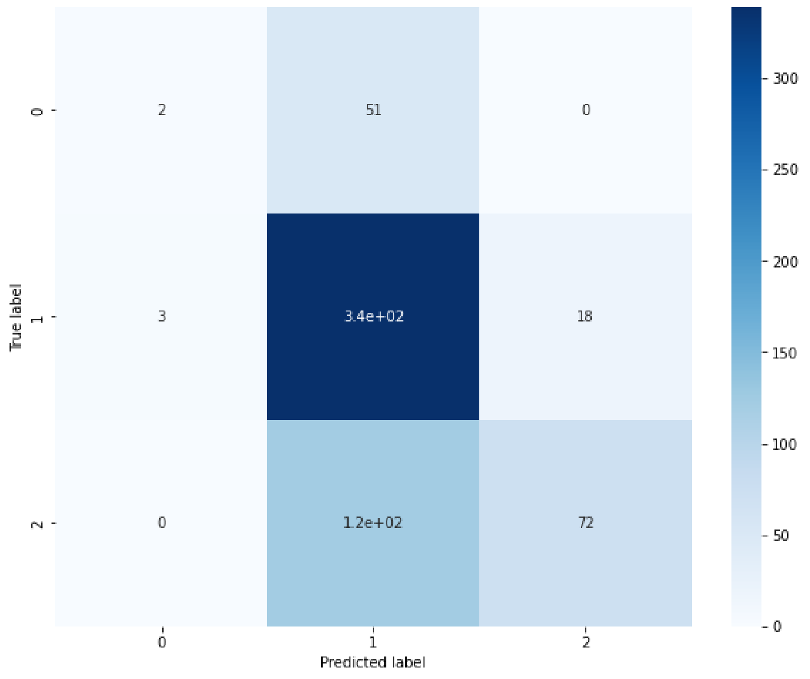
- **Random Forrest :-**



- **SVM:-**



- **Logistic Regression:-**



Conclusion : As for this milestone of the project, All three models worked well when the output class was **Intermediate** but they were weak with class **High** and very weak with class **Low** , and we figured out the reason for this and it is that the Intermediate class contains the biggest part of the data but as for High class it has nearly half of Intermediate class and the Low class has very little part of the data that can easily be neglected.