Sentiment Analysis of movie reviews

Team members (T1):

احمد اسامة عبد الرزاق 20201700020

احمد اديب معضماني 20201701044

الاء مصطفى صادق على 20201700137

أميره ياسر إبراهيم حسين20201700150

شهد اشرف احمد عبداللاه20201700407

مصعب ثابت محمد عبد الآخر20201700848

preprocessing:

❖ At first we split the data into train and test so we can train and test the model , then we clean the data so we can deal with it , so we used some techniques to apply that.it’s 70 % train and 30% test

1. Remove html tags : It's used to identify the html tags in the text then using function sub() to replace the tags with an empty string. It take the characters you want to replace and the characters you want replace with and text.

2. Convert to lower case : It converts all the uppercase characters into lower case characters using function lower().

3. Remove punctuation : It's used to remove punctuation marks from text. Using re.sub() that replaces the punctuation marks with empty string and return the text without punctuation marks.

4. Tokenization: It uses the function word_tokenize() from the library nltk (natural language toolkit) , it’s used to split the text into individual words called tokens .

5. Remove stop words : Stop words is a set of common words like (‘the’ , ‘is’ , ‘and’ , etc) that are often filtered out from the text because they don’t contribute much to the meaning. The stopwords.words(‘English’) function fetches the set of English stop words from nltk’s corpus then we remove them and return the tokens after removing the stop words.

6. Stemming : Stemming maps different forms of the same word to a common “stem”. We used snowball stemmer, it creates a stemmer object for English words . It takes a list of tokens and finds the root of it and returns it .

7. Join the tokens : We use function “ ”.join() that takes list of tokens and join them into a single string. We use single space as a separator , so each token will be separated by a space in the resulting string .

8. Standard scaling : It standardizes a feature by subtracting the mean and then scaling to unit variance, making sure they have a mean of approximately 0 and a standard deviation of approximately 1. we use it because it makes the data more easier to deal with.

9. Maxabs scaling : it Scales each feature by its maximum absolute value.by subtracting the minimum value in the feature and then dividing by the range by MaxAbsScaler() .

We used both scalers because it provides a more comprehensive normalization of the data, potentially improving the model's ability to learn and enhance the robustness of the preprocessing step

10. Reduce dimensions of data : it’s used to reduce the dimensionalty of the feature space by using singular value decomposition (svd) and it operates efficiently on sparse matrix and number of components = 200 , TruncatedSVD() function.

11. Label encoding : it’s used to encode categorical labels into numerical ones by using LabelEncoder() , it’s being trained by the train data.

Feature Extraction

We used tf-idf vectorizer for feature extraction

It converts text documents into numerical feature vectors based on the frequencies of terms (words or n-grams) in the documents while also considering the importance of these terms across the entire corpus. It returns sparse matrix.

Term Frequency (TF): It computes the frequency of each term (word or n-gram) within each document. Terms that occur more frequently within a document are assigned higher weights.

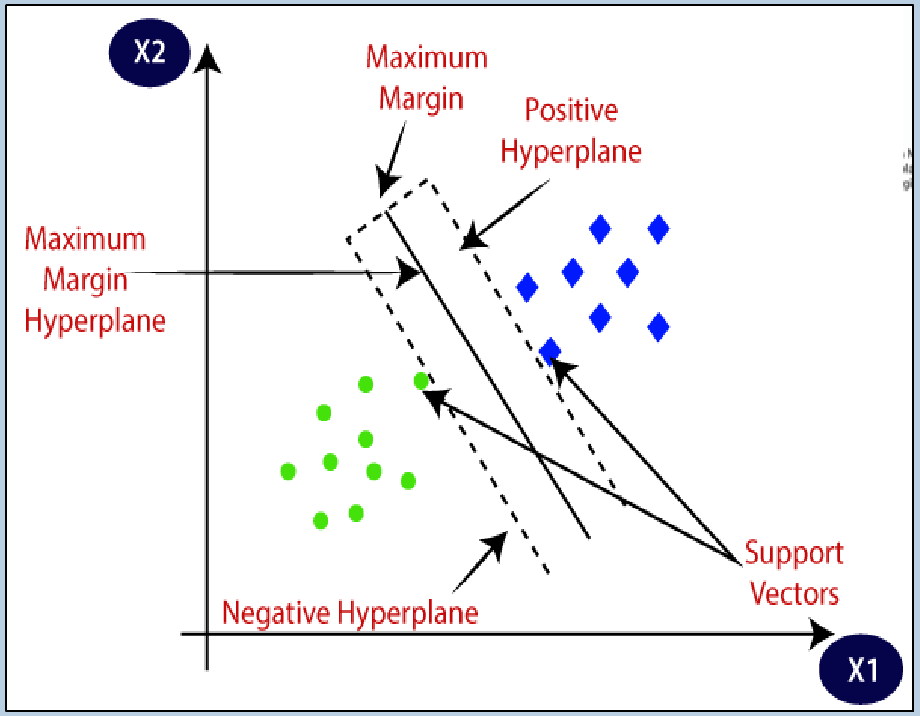
Inverse Document Frequency (IDF): It computes the inverse document frequency of each term across the entire corpus. Terms that occur frequently across many documents are assigned lower weights.

TF-IDF Weighting: It combines the TF and IDF scores to compute the final TF-IDF weight for each term in each document. Terms that are frequent in a document but rare across the corpus are given higher weights, indicating their importance in distinguishing that document from others.

Model training and testing:

❖ **SVM (support vector machine)**

The main objective of the SVM algorithm is to find the optimal hyperplane in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features.



Model training and testing:

❖ SVM (support vector machine)

1. Hyper parameters tuning:

We use gridsearchcv for finding the optimal parameter values from a given set of parameters in a grid and parameters it takes : cv: This parameter specifies the number of folds for cross-validation and the param grid

Param_grid defines a directory of the hyperparameters to tune which are : “c “(regularization parameter) , “gamma” (kernel cofficient)

C: The C hyperparameter controls the regularization strength in SVM. It balances between maximizing the margin and minimizing the classification error. A smaller C leads to a larger margin but may allow for more misclassifications on the training data, while a larger C penalizes misclassifications more heavily, potentially resulting in a smaller margin.

gamma: The gamma hyperparameter defines the kernel coefficient. It influences the influence of each training example on the decision boundary. A small gamma value means a large influence, leading to smoother decision boundaries, while a large gamma value means a smaller influence, resulting in more complex decision boundaries.

Model training and testing:

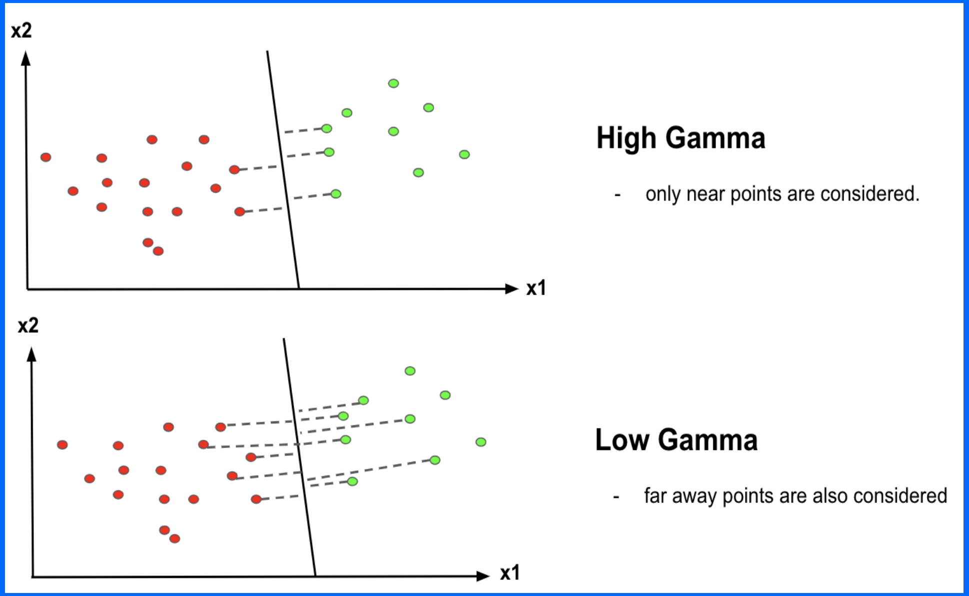
❖ SVM (support vector machine)

2.Training the model

We fit the model after getting the best hyper parameters of the grid search and train the model that takes x train and y train data

We train the model to make it learn the data and to perform prediction on new unseen data.then we calculate the train accuracy.

```
gridsearch.fit( x_train , y_train)
```


x2
x1
High Gamma
- only near points are considered.
x2
x1
Low Gamma
- far away points are also considered

Model training and testing:

❖ **SVM (support vector machine)**

3.testing the model

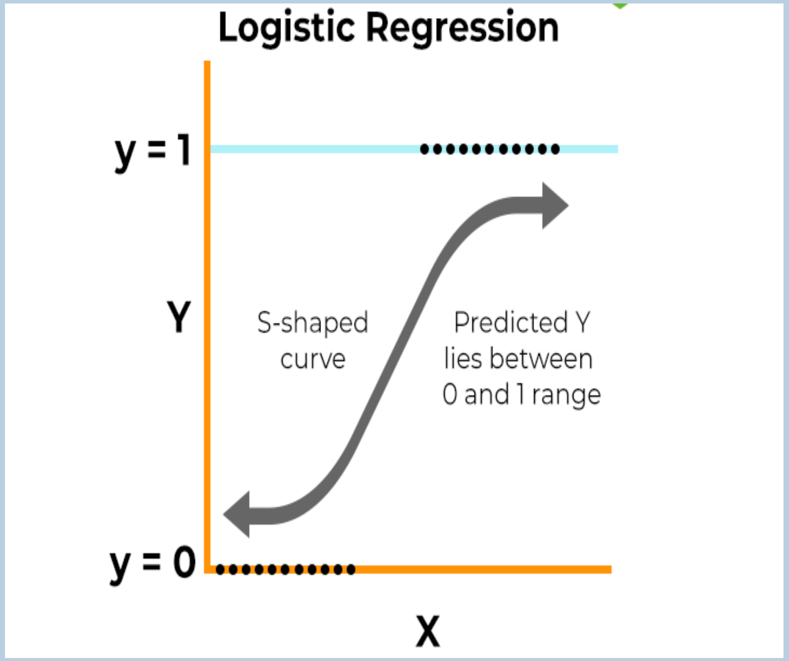
After training the model predictions can be made on the test data. The predict method of the SVM model is used to predict the class labels or decision function values for the test instances.

Evaluation: After making predictions on the test data, the performance of the SVM model is evaluated using appropriate evaluation metrics. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score and Roc Auc.These metrics provide insights into how well the model generalizes to unseen data.

Model training and testing:

❖ **Logistic Regression**

a statistical model that models the log-odds of an event as a linear combination of one or more independent variables. Logistic regression is a supervised machine learning algorithm widely used for binary classification tasks



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

Model training and testing:

❖ Logistic Regression

1. Hyper parameters tuning:

We use gridsearchcv for finding the optimal parameter values from a given set of parameters in a grid and parameters it takes : cv: the number of folds for cross-validation and the param grid.

Param_grid defines a directory of the hyperparameters to tune which are : “c “(regularization parameter) , “penalty” , “solver”

'C': Regularization parameter. It controls the inverse of regularization strength, where smaller values specify stronger regularization.

'penalty': Type of regularization. It specifies the norm used in the penalization. 'l1' refers to L1 regularization (Lasso), and 'l2' refers to L2 regularization (Ridge).

'solver': Algorithm to use in the optimization problem. 'saga' supports both L1 and L2 regularization.

Model training and testing:

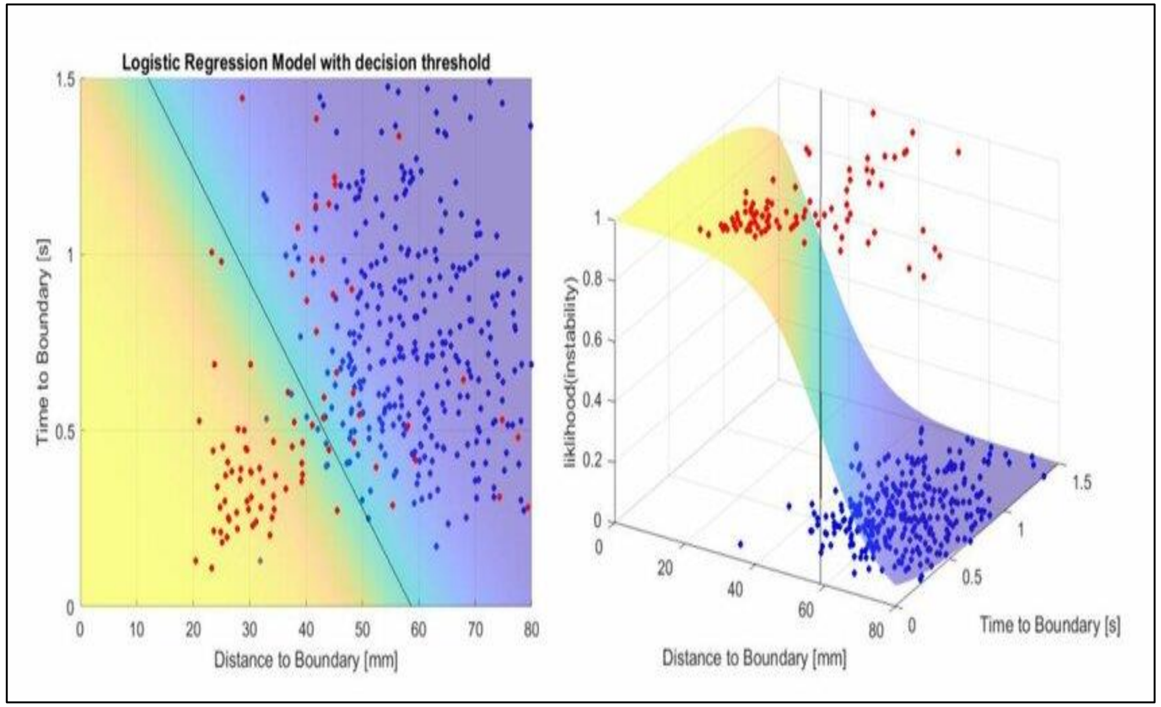
❖ Logistic Regression

2.Training the model

We fit the model after getting the best hyper parameters of the grid search and train the model that takes x train and y train data

We train the model to make it learn the data and to perform prediction on new unseen data.then we calculate the train accuracy.

```
gridsearch.fit( x_train , y_train)
```



Model training and testing:

❖ **Logistic Regression**

3.testing the model

After training the model predictions can be made on the test data. The predict method of the SVM model is used to predict the class labels or decision function values for the test instances.

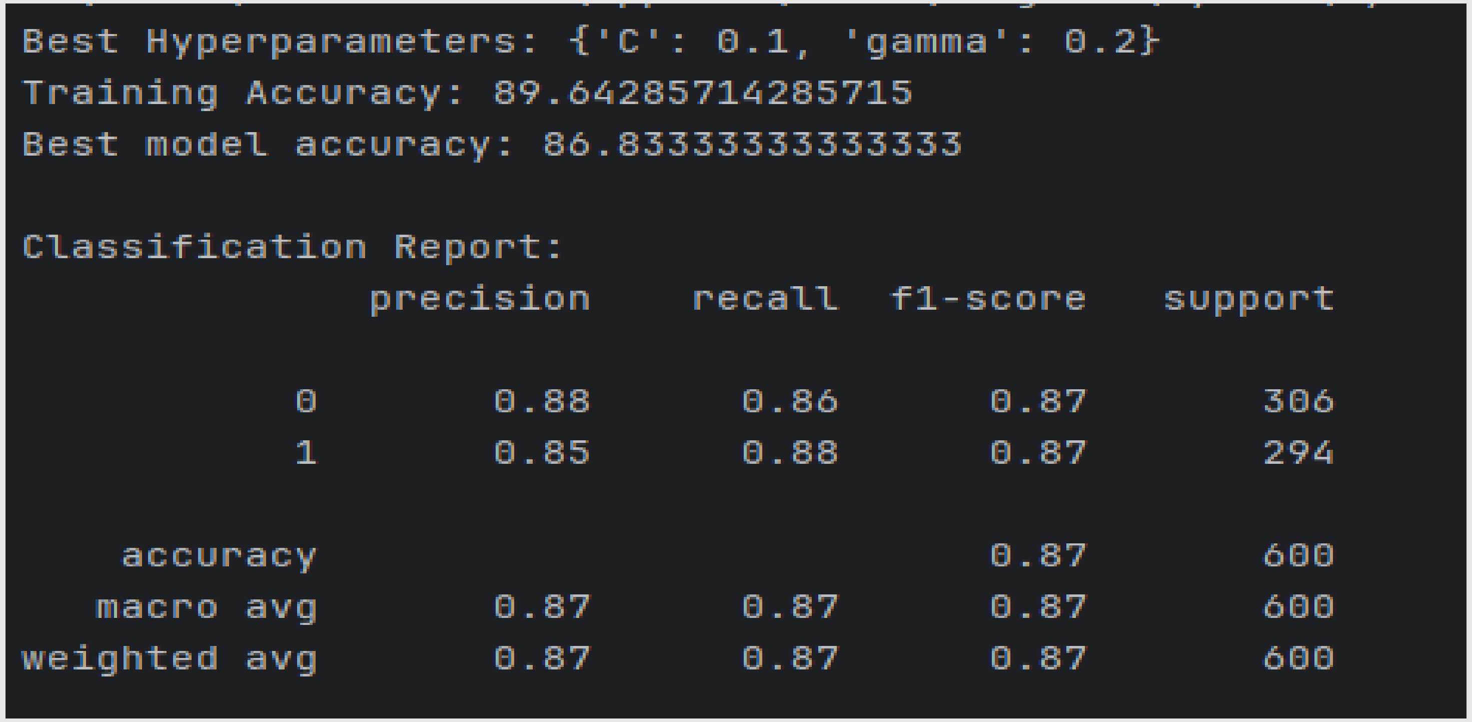
Evaluation: After making predictions on the test data, the performance of the SVM model is evaluated using appropriate evaluation metrics. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score and Roc Auc.These metrics provide insights into how well the model generalizes to unseen data.

Visualizing results

❖ SVM (support vector machine)

we got accuracy, precision, recall, F1-score and Roc Auc of this model

❖ This is the best accuracy we got

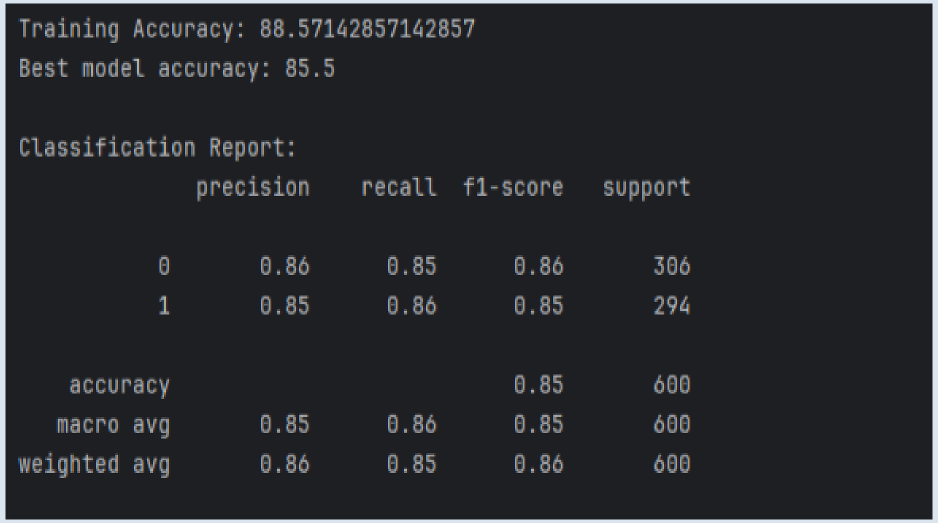


```
Best Hyperparameters: {'C': 0.1, 'gamma': 0.2}
Training Accuracy: 89.64285714285715
Best model accuracy: 86.83333333333333

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       306
           1       0.85      0.88      0.87       294

    accuracy                           0.87       600
   macro avg       0.87      0.87      0.87       600
weighted avg       0.87      0.87      0.87       600
```

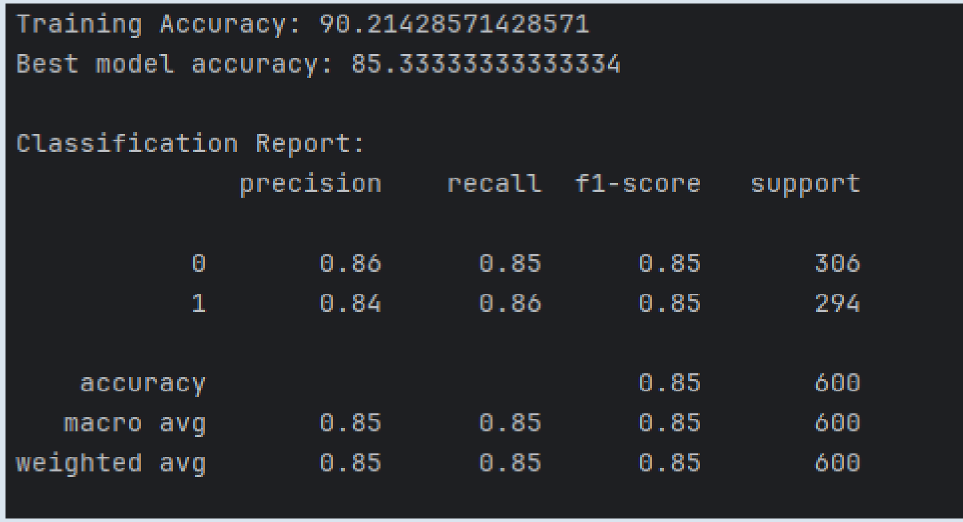
Visualizing results

❖ SVM (support vector machine)

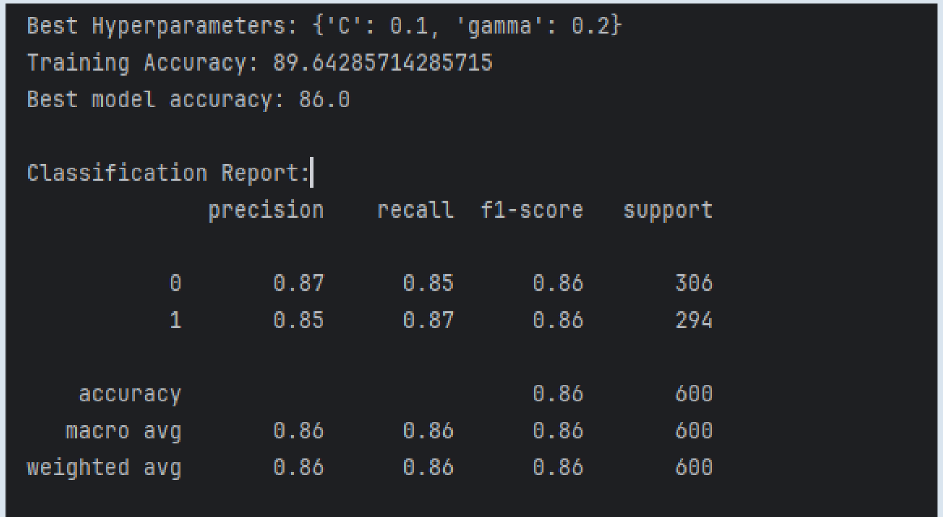
❖ Here are some screenshots from our model due to the random state



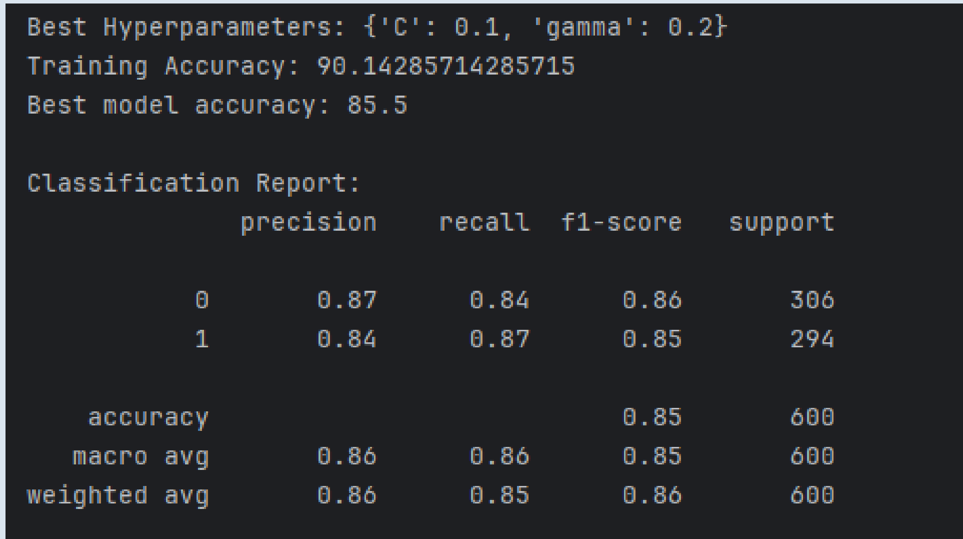
```
Training Accuracy: 88.57142857142857
Best model accuracy: 85.5

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.85      0.86       306
           1       0.85      0.86      0.85       294

    accuracy                           0.85       600
   macro avg       0.85      0.86      0.85       600
weighted avg       0.86      0.85      0.86       600
```



```
Training Accuracy: 90.21428571428571
Best model accuracy: 85.33333333333334

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.85      0.85       306
           1       0.84      0.86      0.85       294

    accuracy                           0.85       600
   macro avg       0.85      0.85      0.85       600
weighted avg       0.85      0.85      0.85       600
```



```
Best Hyperparameters: {'C': 0.1, 'gamma': 0.2}
Training Accuracy: 89.64285714285715
Best model accuracy: 86.0

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       306
           1       0.85      0.87      0.86       294

    accuracy                           0.86       600
   macro avg       0.86      0.86      0.86       600
weighted avg       0.86      0.86      0.86       600
```



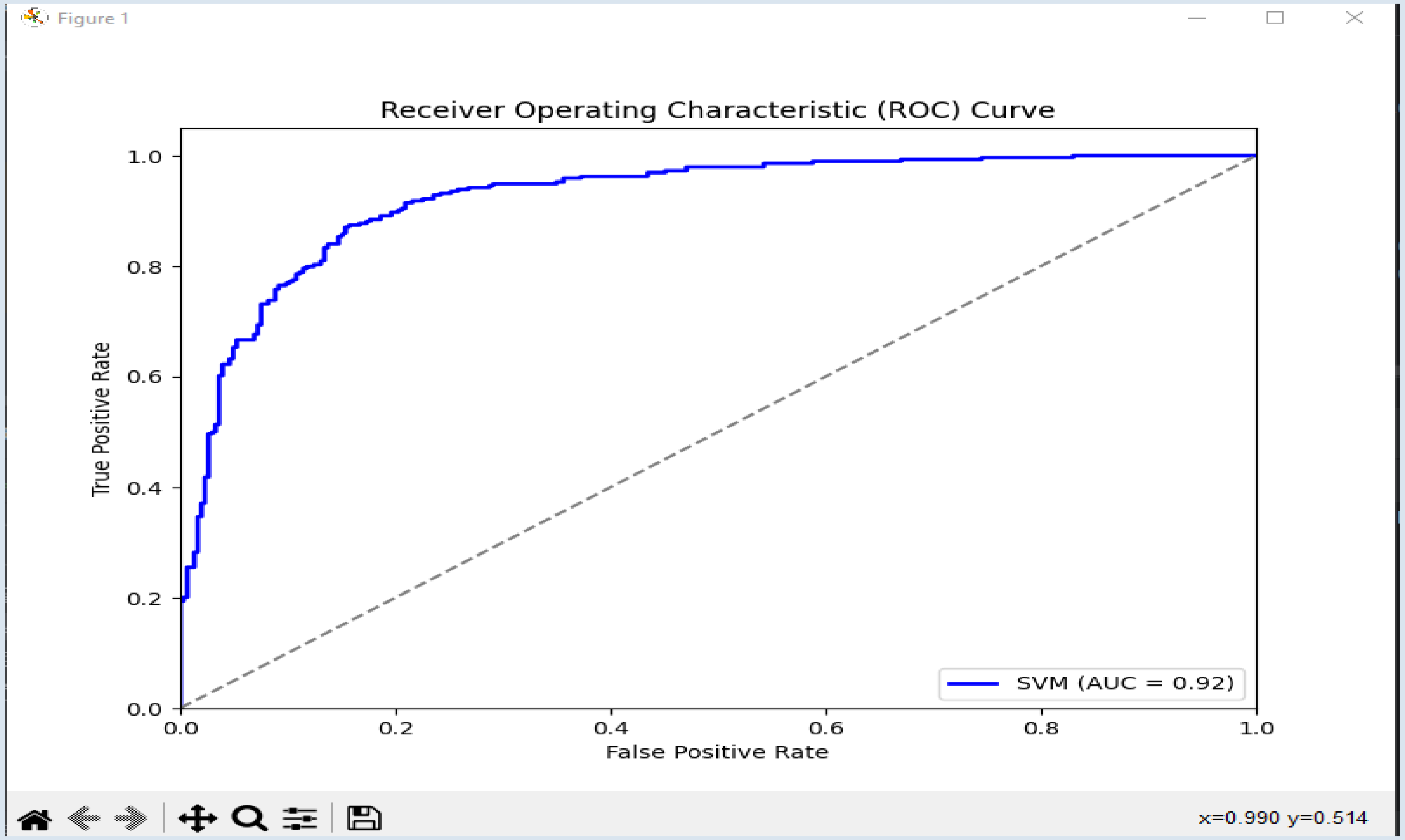
```
Best Hyperparameters: {'C': 0.1, 'gamma': 0.2}
Training Accuracy: 90.14285714285715
Best model accuracy: 85.5

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.84      0.86       306
           1       0.84      0.87      0.85       294

    accuracy                           0.85       600
   macro avg       0.86      0.86      0.85       600
weighted avg       0.86      0.85      0.86       600
```

Visualizing results

❖ **SVM (support vector machine)**

- **Roc Auc of model svm :is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:**
  - ➢ **True Positive Rate**
  - ➢ **False Positive Rate**

Our model AUC = 0.92

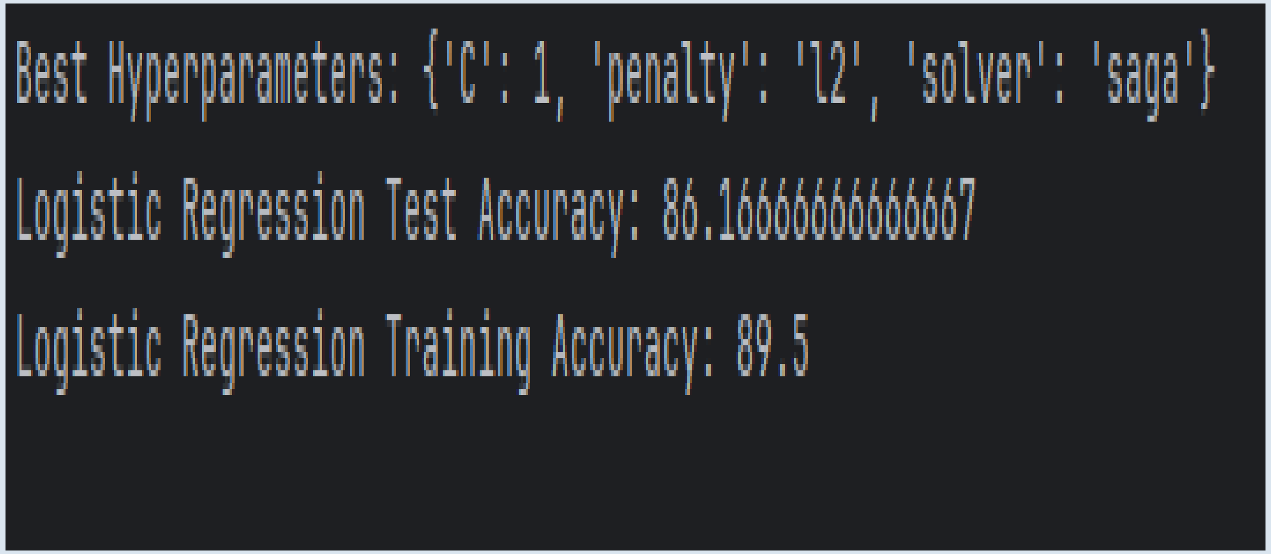


Visualizing results

❖ Logistic Regression

we got accuracy, precision, recall, F1-score and Roc Auc of this model

- ❖ This is the best accuracy we got

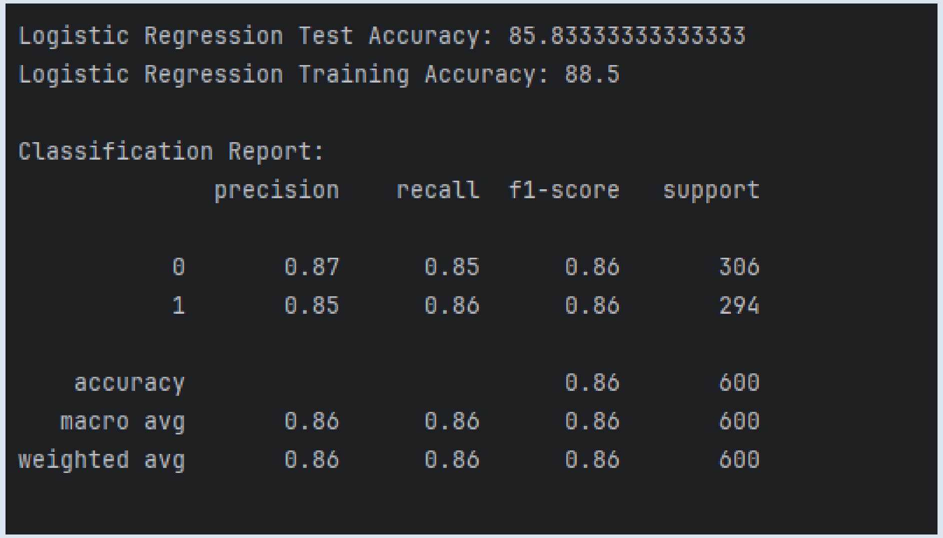


```
Best Hyperparameters: {'C': 1, 'penalty': 'l2', 'solver': 'saga'}
Logistic Regression Test Accuracy: 86.16666666666667
Logistic Regression Training Accuracy: 89.5
```

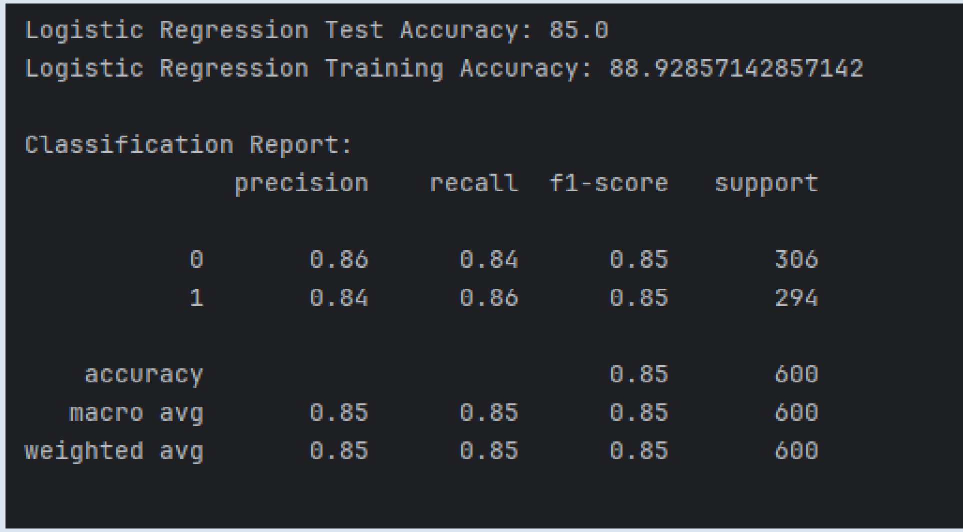
Visualizing results

❖ SVM (support vector machine)

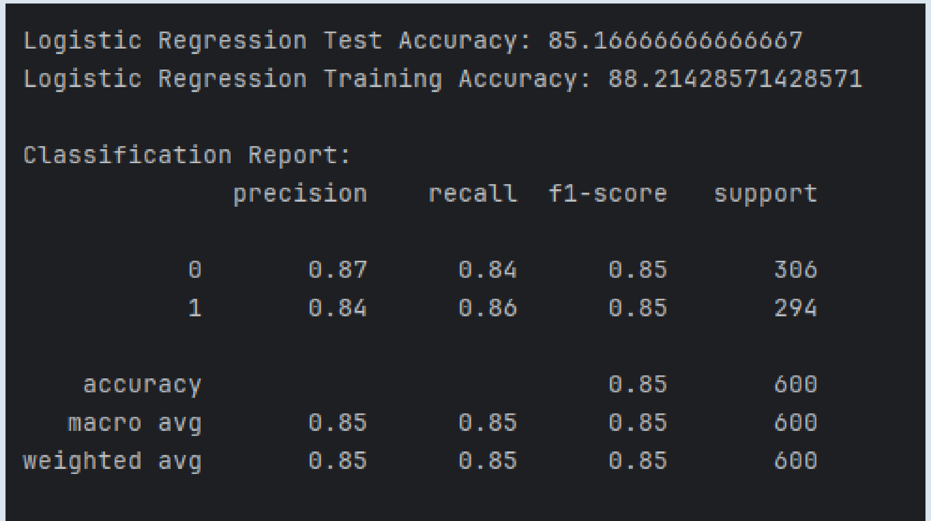
❖ Here are some screenshots from our model due to the random state



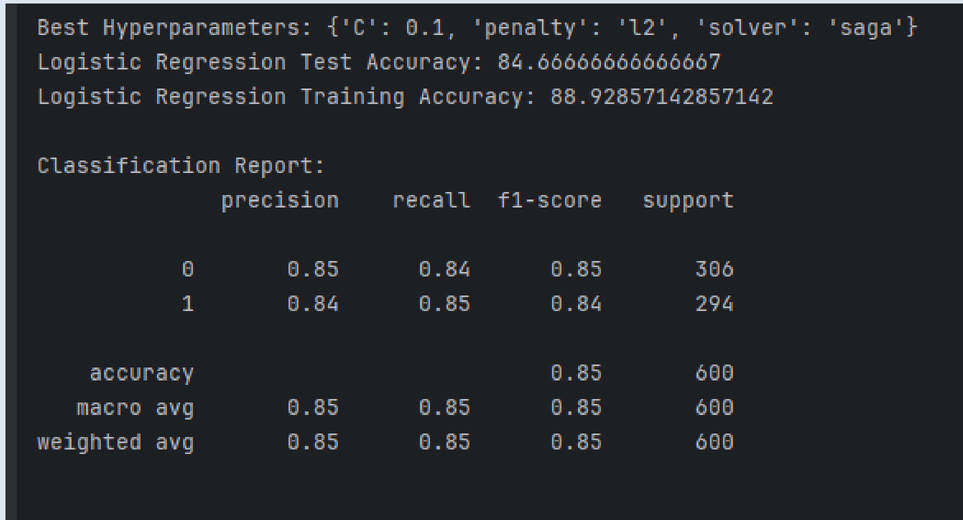
```
Logistic Regression Test Accuracy: 85.83333333333333
Logistic Regression Training Accuracy: 88.5

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       306
           1       0.85      0.86      0.86       294

    accuracy                           0.86       600
   macro avg       0.86      0.86      0.86       600
weighted avg       0.86      0.86      0.86       600
```



```
Logistic Regression Test Accuracy: 85.0
Logistic Regression Training Accuracy: 88.92857142857142

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.84      0.85       306
           1       0.84      0.86      0.85       294

    accuracy                           0.85       600
   macro avg       0.85      0.85      0.85       600
weighted avg       0.85      0.85      0.85       600
```



```
Logistic Regression Test Accuracy: 85.16666666666667
Logistic Regression Training Accuracy: 88.21428571428571

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.84      0.85       306
           1       0.84      0.86      0.85       294

    accuracy                           0.85       600
   macro avg       0.85      0.85      0.85       600
weighted avg       0.85      0.85      0.85       600
```



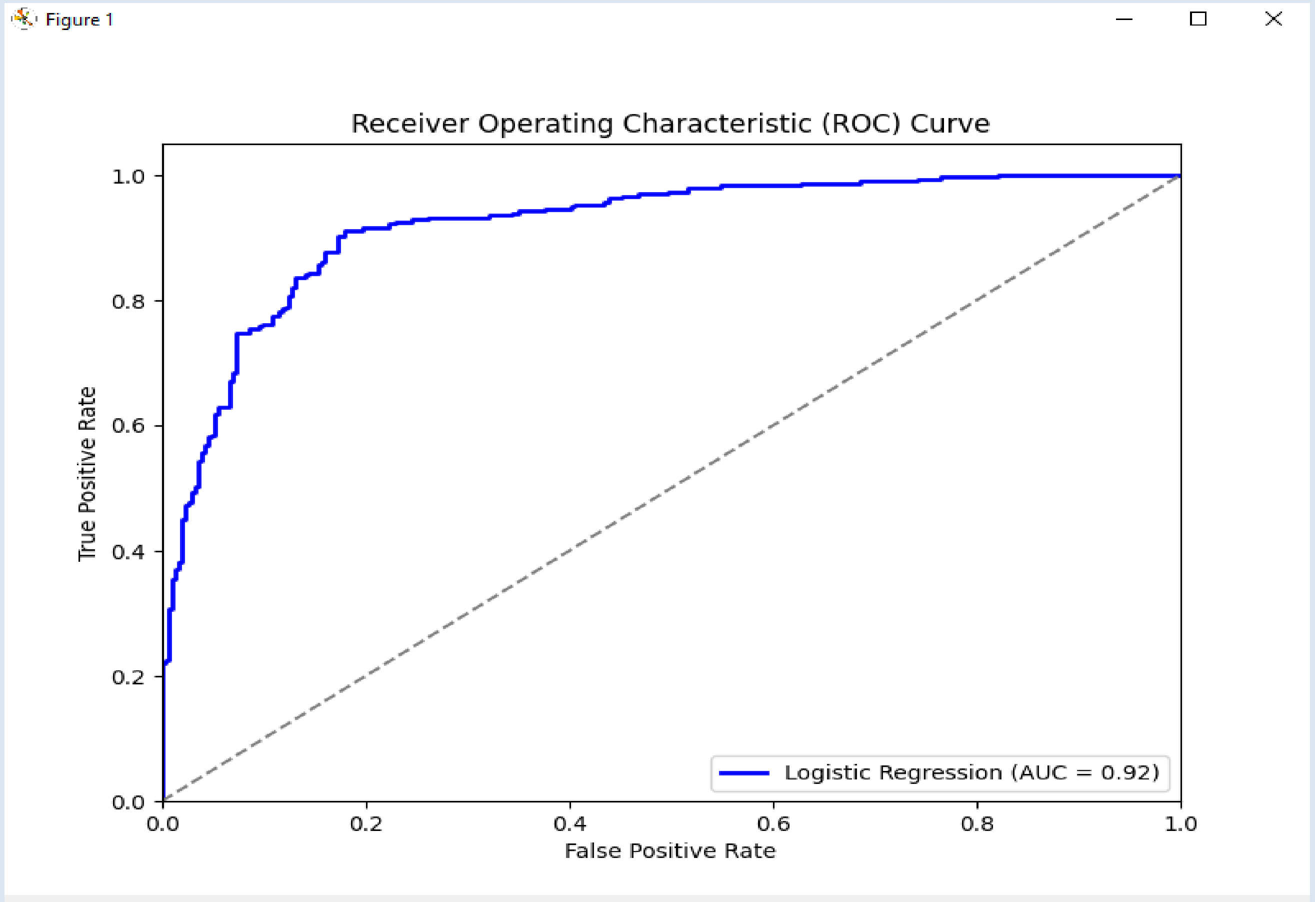
```
Best Hyperparameters: {'C': 0.1, 'penalty': 'l2', 'solver': 'saga'}
Logistic Regression Test Accuracy: 84.66666666666667
Logistic Regression Training Accuracy: 88.92857142857142

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85       306
           1       0.84      0.85      0.84       294

    accuracy                           0.85       600
   macro avg       0.85      0.85      0.85       600
weighted avg       0.85      0.85      0.85       600
```

Visualizing results

❖ Logistic Regression

- Roc Auc of model svm :is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:
  - ➢ True Positive Rate
  - ➢ False Positive Rate

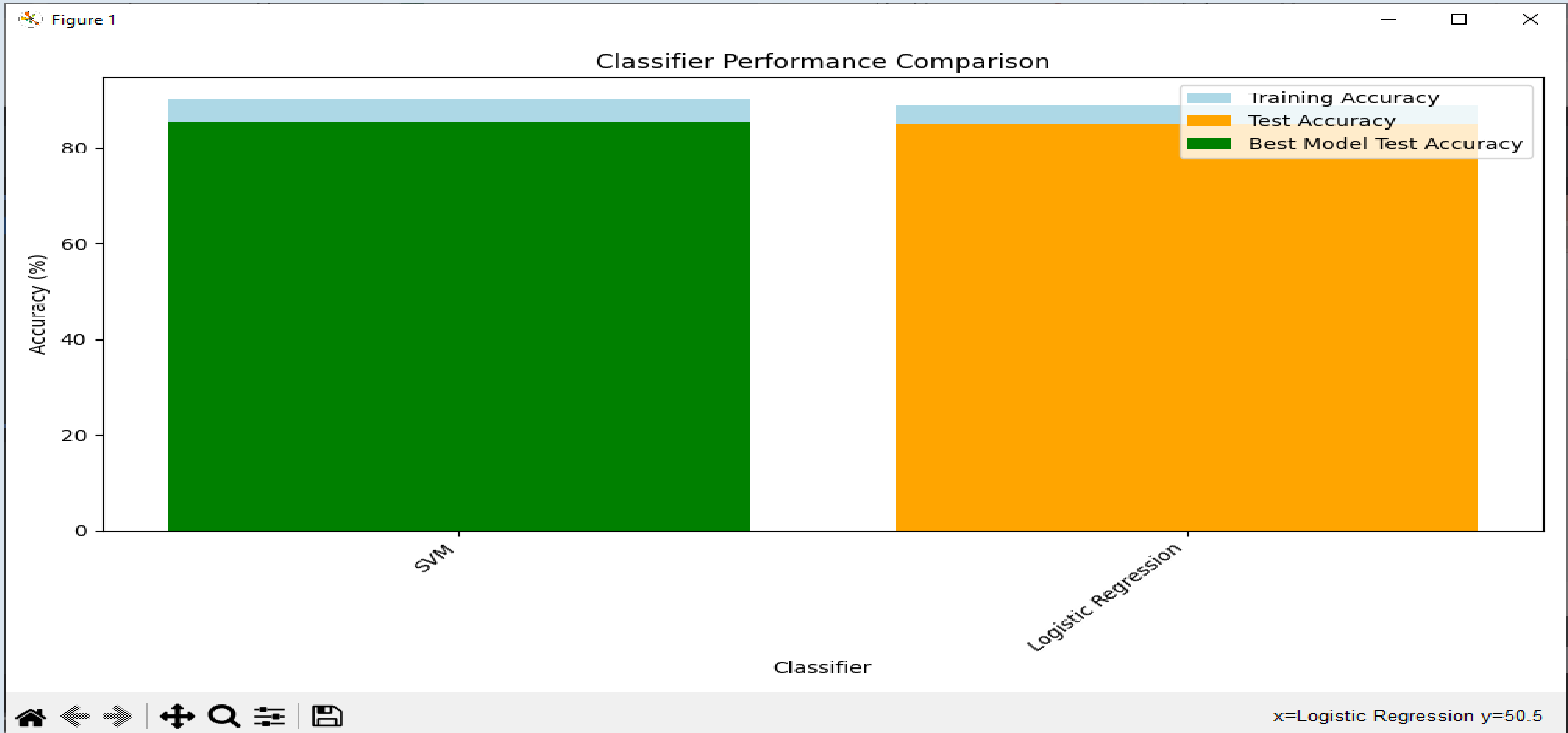
Our model AUC = 0.92



Visualizing results

- ❖ We compared the models based on the accuracuy and we got an conclusion that svm is better than logistic for this data

- ❖ Green here reference to the best model in test accuracy



Saving the models

❖ We saved the models we trained by joblib to load it and test it to see how the models learned from the data

❖ Models like

- TfidfVectorizer()
- StandardScaler()
- MaxAbsScaler()
- TruncatedSVD()
- LabelEncoder()
- Svm
- Logistic regression

Thank you