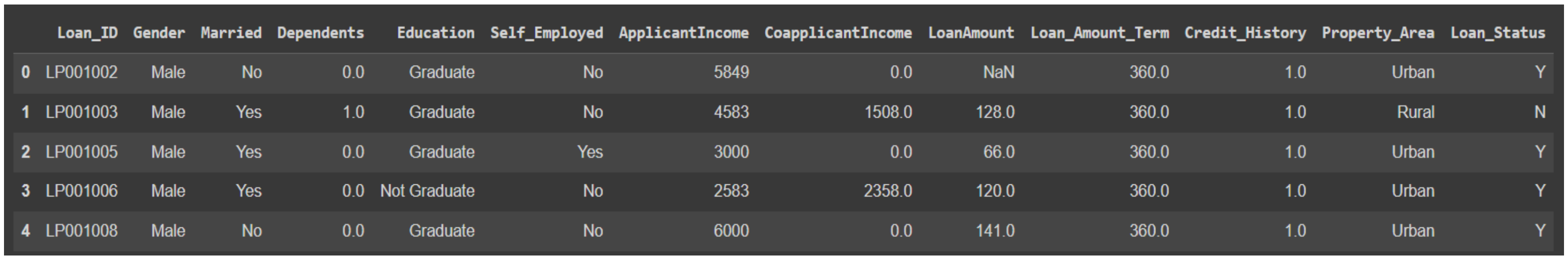
IML PROJECT REPORT

Aim of the project

Based on an individual's particular features, we aim to devise a model that accurately predicts the loan status of the individual. (Binary classification)

Following is the dataset we will be using:

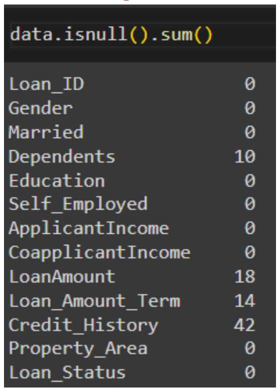


	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

So, based on a person's gender, marital status, number of people dependent, employment status, etc, we need to tell if the person gets the loan or not.

Data Preprocessing- Before proceeding with model training and selection, it is essential to pre-process the data. Here are the steps we followed for the same:

1. Handling the missing values:



```
data.isnull().sum()

Loan_ID               0
Gender                0
Married               0
Dependents           10
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           18
Loan_Amount_Term     14
Credit_History       42
Property_Area         0
Loan_Status           0
```

- The categorical column - 'Loan Amount term,' which had missing values, was filled by the column's mode.

  Why? - According to us, the given dataset is a sample from a larger population; filling the missing Loan Amount term with the mode is a

reasonable assumption that the most common value will likely represent the broader population. So, this approach is better when the model is deployed to handle new, unseen data.

- The numerical column ‘Loan Amount’s’ missing values were filled with the mean of the column.

  Why? - Similar to the above reason, we can assume that a person’s loan amount is around the average of the whole population to maintain the statistical power of our analysis.

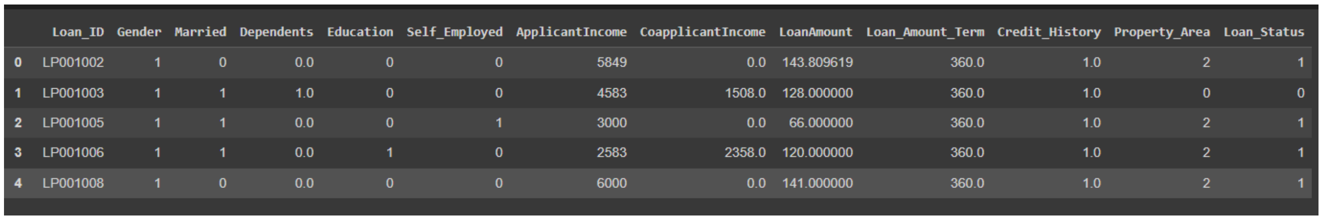
- The numerical column ‘Credit History’s’ missing values were filled with 0.

  Why? - We do not have prior knowledge of an individual’s credit history, so it is better to assume that he had none than anything else.

- The categorical column - ‘dependants,’ which had missing values, was filled by defining a separate function so that if the person was married, its dependent column was filled with 1; else 0.

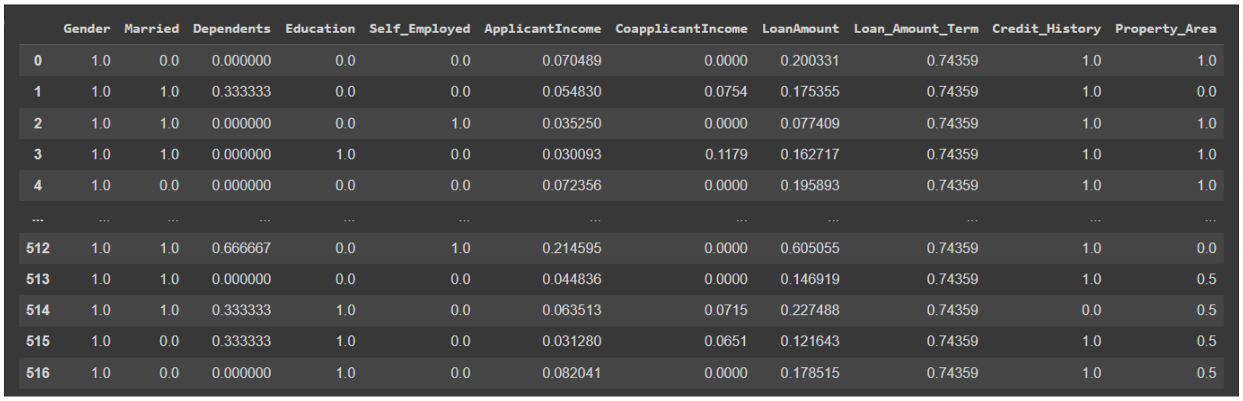
  Why? - Given that an individual is married, it can be said for sure that at least one person is dependent on them.

2. Label Encoding:
   Here, the categorical columns, including the binary and ordinal columns, were label-encoded. The dataset now becomes:



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	1	0	0.0	0	0	5849	0.0	143.809619	360.0	1.0	2	1
1	LP001003	1	1	1.0	0	0	4583	1508.0	128.000000	360.0	1.0	0	0
2	LP001005	1	1	0.0	0	1	3000	0.0	66.000000	360.0	1.0	2	1
3	LP001006	1	1	0.0	1	0	2583	2358.0	120.000000	360.0	1.0	2	1
4	LP001008	1	0	0.0	0	0	6000	0.0	141.000000	360.0	1.0	2	1

3. Normalizing the data:
   The columns excluding the target column (Loan Status) and one other column (‘Loan ID’ - no sense in normalizing the person's ID as it will not be considered while classifying) were normalized.
   The normalized dataset looks like this:

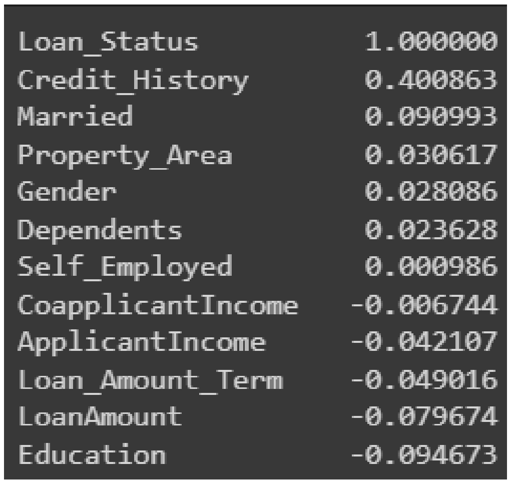


	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
0	1.0	0.0	0.000000	0.0	0.0	0.070489	0.0000	0.200331	0.74359	1.0	1.0
1	1.0	1.0	0.333333	0.0	0.0	0.054830	0.0754	0.175355	0.74359	1.0	0.0
2	1.0	1.0	0.000000	0.0	1.0	0.035250	0.0000	0.077409	0.74359	1.0	1.0
3	1.0	1.0	0.000000	1.0	0.0	0.030093	0.1179	0.162717	0.74359	1.0	1.0
4	1.0	0.0	0.000000	0.0	0.0	0.072356	0.0000	0.195893	0.74359	1.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...
512	1.0	1.0	0.666667	0.0	1.0	0.214595	0.0000	0.605055	0.74359	1.0	0.0
513	1.0	1.0	0.000000	0.0	0.0	0.044836	0.0000	0.146919	0.74359	1.0	0.5
514	1.0	1.0	0.333333	1.0	0.0	0.063513	0.0715	0.227488	0.74359	0.0	0.5
515	1.0	0.0	0.333333	1.0	0.0	0.031280	0.0651	0.121643	0.74359	1.0	0.5
516	1.0	0.0	0.000000	1.0	0.0	0.082041	0.0000	0.178515	0.74359	1.0	0.5

4. Exploratory Data Analysis:

- **Correlation Matrix** - To better visualize the dataset, we calculated the correlation matrix, which tells us how the target feature, i.e., 'Loan Status,' depends on other features like marital status, loan amount, credit history, property area, etc.

  The results we got are:

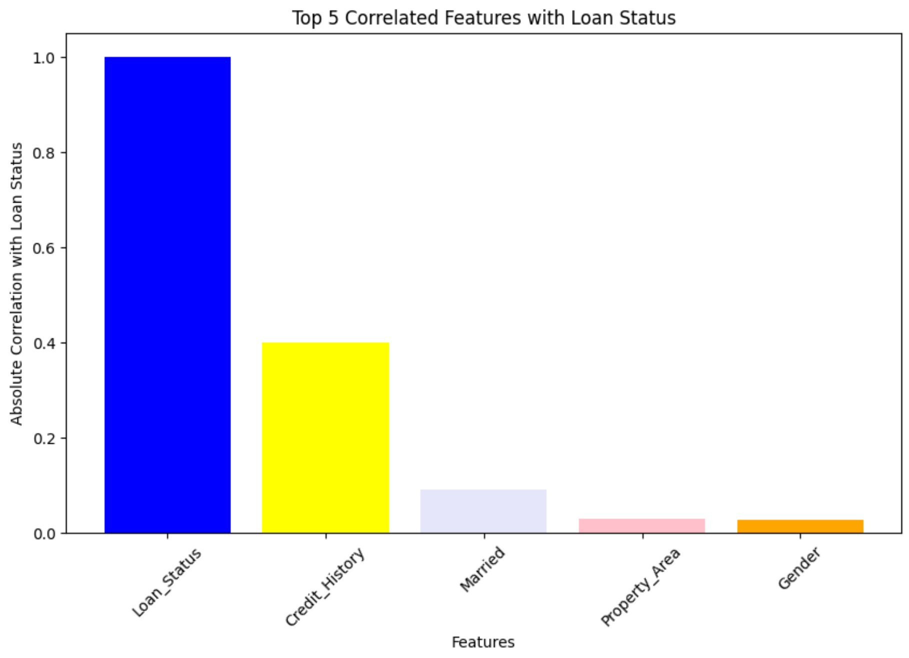


```
Loan_Status          1.000000
Credit_History       0.400863
Married              0.090993
Property_Area        0.030617
Gender               0.028086
Dependents           0.023628
Self_Employed        0.000986
CoapplicantIncome   -0.006744
ApplicantIncome     -0.042107
Loan_Amount_Term    -0.049016
LoanAmount          -0.079674
Education           -0.094673
```

- So we can infer that whether the person is getting a loan or not majorly depends on his credit history, with a positive correlation of 40%.

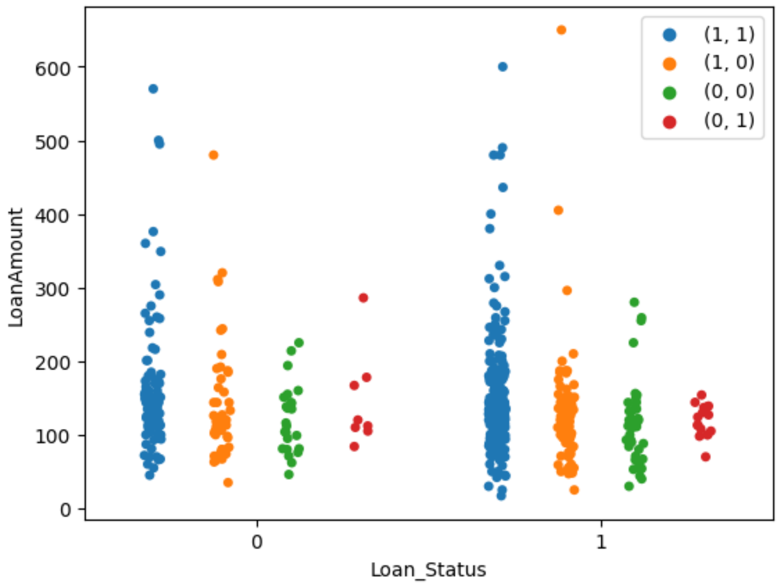
- It can also be noted that features like Loan amount have a negative correlation value, which means that as Loan amount increases, the probability of the person getting the loan tends to decrease.

- **Graph of top 5 correlated features with loan status to visualize the above result-**



Order of dependence of Loan status -
Credit History>Married>Property Area>Gender.

- **Stripplot between Loan status and Loan Amount considering two features - Gender and marriage.**

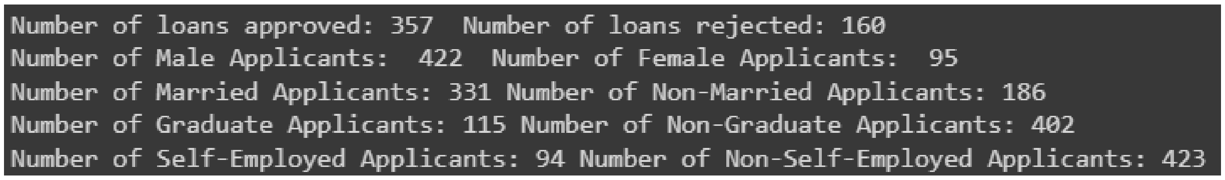


So, for example, from here, we can infer that given a certain amount of loan that a person wants and his/her gender and marital status, will he/she get the loan or not?

Observations:

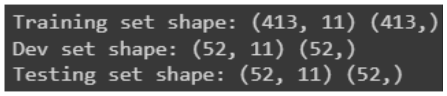
Married, male, amount<200 = highest probability of approval
Married, female, amount>150 = lowest probability of approval

- **Stats -**



```
Number of loans approved: 357  Number of loans rejected: 160
Number of Male Applicants:  422  Number of Female Applicants:  95
Number of Married Applicants: 331 Number of Non-Married Applicants: 186
Number of Graduate Applicants: 115 Number of Non-Graduate Applicants: 402
Number of Self-Employed Applicants: 94 Number of Non-Self-Employed Applicants: 423
```

Train Test Split -
We first divided the dataset into train and temp and then divided the temp set into test and dev sets. So finally, we have three sets -
Train - 80% of the entire dataset
Test - 10% of the entire dataset
Dev - 10% of the entire dataset

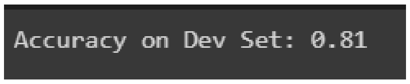


```
Training set shape: (413, 11) (413,)
Dev set shape: (52, 11) (52,)
Testing set shape: (52, 11) (52,)
```

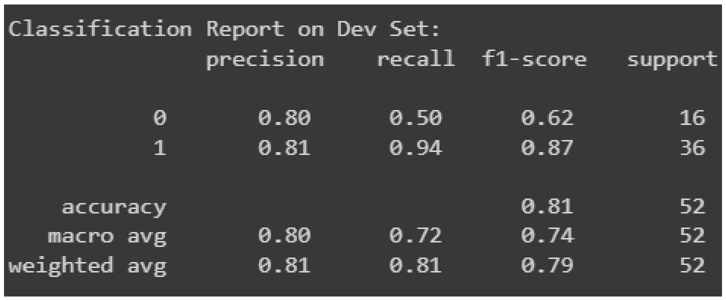
Model training and selection-
We trained three classification models - SVM, Decision trees, and Logistic Regression, on the training set and found their accuracies in classifying on the dev set. The model with the best accuracy will be chosen, and then predictions and accuracy will be calculated on the test set, which was untouched until now. This way, we will select and evaluate the best model for our classification task and evaluate it.

1. SVM
Three SVM models - linear, polynomial, and RBF were implemented, and their respective evaluation was done to select the best.

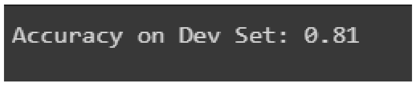
- Linear SVM:

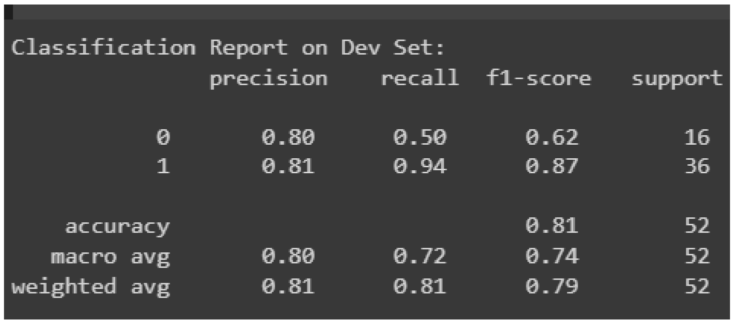


```
Accuracy on Dev Set: 0.81
```

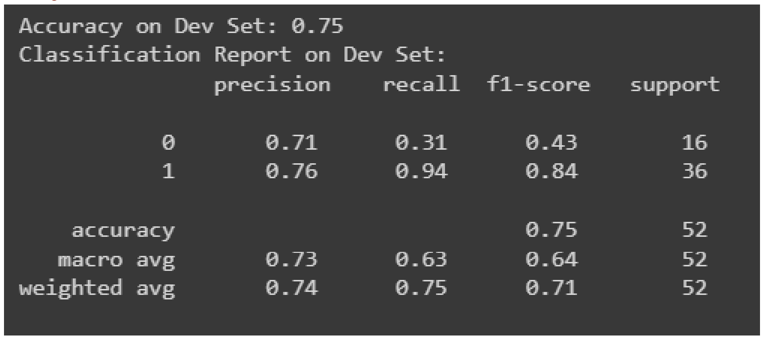

```
Classification Report on Dev Set:
              precision    recall  f1-score   support

           0       0.80      0.50      0.62        16
           1       0.81      0.94      0.87        36

    accuracy                           0.81        52
   macro avg       0.80      0.72      0.74        52
weighted avg       0.81      0.81      0.79        52
```

- RBF kernel:


```
Accuracy on Dev Set: 0.81
```


```
Classification Report on Dev Set:
              precision    recall  f1-score   support

           0       0.80      0.50      0.62        16
           1       0.81      0.94      0.87        36

    accuracy                           0.81        52
   macro avg       0.80      0.72      0.74        52
weighted avg       0.81      0.81      0.79        52
```

- Polynomial kernel:


```
Accuracy on Dev Set: 0.75
Classification Report on Dev Set:
              precision    recall  f1-score   support

           0       0.71      0.31      0.43        16
           1       0.76      0.94      0.84        36

    accuracy                           0.75        52
   macro avg       0.73      0.63      0.64        52
weighted avg       0.74      0.75      0.71        52
```

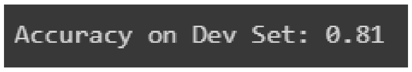
Here, we observe that the accuracy for polynomial SVM is the lowest (75%). For linear and RBF, the accuracy and other metrics are the same (81%).

So now, in order to choose between the latter two models, we perform hyperparameter tuning using GridSearchCV from scikit-learn and then compare the model's performance based on the best hyperparameters.

Hyperparameter tuning for linear SVM -

GridSearchCV for hyperparameter tuning was used. Different values of C were tried, and the combination that resulted in the best cross-validated performance was selected.

A new SVM model was created with the best identified hyperparameters, and then the model's accuracy was calculated. The result was:



```
Accuracy on Dev Set: 0.81
```

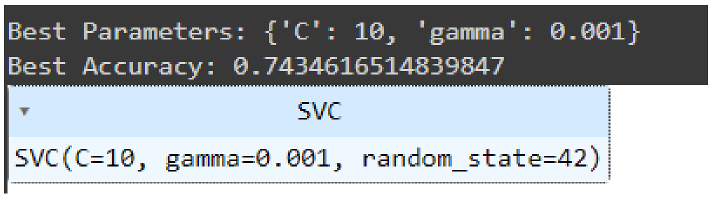
Here, the regularization parameter - 'C' can be understood as:

- Smaller C values mean a simpler model (smooth decision boundary) that may generalize better but may not fit the training data as closely.
- Larger C values lead to a more complex model (complex, intricate decision boundary) that fits the training data more closely but may not generalize well to unseen data.

The goal was to strike a good balance and find a C that provides good generalization to new, unseen data while fitting the training data well.

Hyperparameter tuning for RBF SVM -

Again, the same method was used for hyperparameter tuning. The result was:



```
Best Parameters: {'C': 10, 'gamma': 0.001}
Best Accuracy: 0.7434616514839847
▾                    SVC
SVC(C=10, gamma=0.001, random_state=42)
```

Here, the gamma parameter can be understood as:

- Small gamma leads to a more global and smoother decision boundary.
- Large gamma leads to a more localized and complex decision boundary.

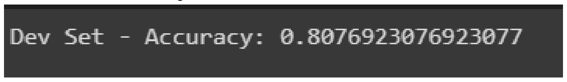
Since our dataset wasn't that complex, a small gamma value is more appropriate to avoid overfitting.

- Comparing the accuracy of both the above SVM models, we can see that it was more for linear SVM (81% > 74%), and hence, we will choose LinearSVC among the two.

2. LOGISTIC REGRESSION

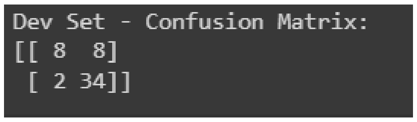
Logistic regression is a simple yet powerful algorithm well-suited for binary classification.

- The accuracy on dev set of this model was:



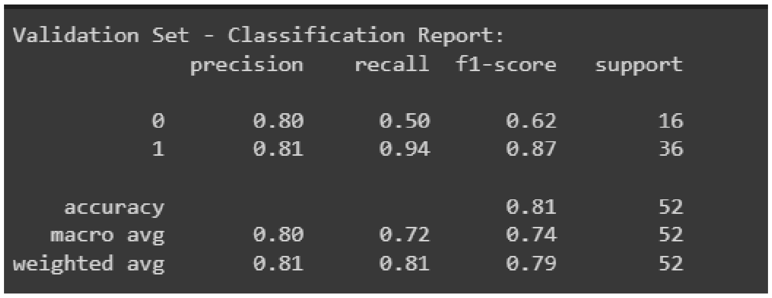
```
Dev Set - Accuracy: 0.8076923076923077
```

- Other performance metrics were:



```
Dev Set - Confusion Matrix:
[[ 8  8]
 [ 2 34]]
```

  - **True Positive (TP)**: 34 instances were correctly predicted as the positive class.
  - **True Negative (TN)**: 8 instances were correctly predicted as the negative class.
  - **False Positive (FP)**: 8 instances were incorrectly predicted as the positive class.
  - **False Negative (FN)**: 2 instances were incorrectly predicted as the negative class.

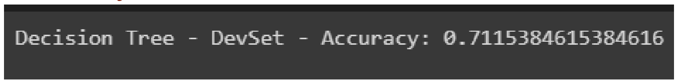


```
Validation Set - Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.50      0.62        16
           1       0.81      0.94      0.87        36

    accuracy                           0.81        52
   macro avg       0.80      0.72      0.74        52
weighted avg       0.81      0.81      0.79        52
```

3. Decision Trees

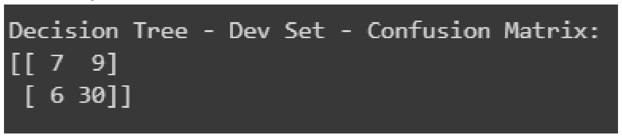
“Decision trees” was the third model for which we calculated the accuracy. We chose this as it provides a visual representation of the decision-making process and can automatically select important features.

- Accuracy on dev set of this model:

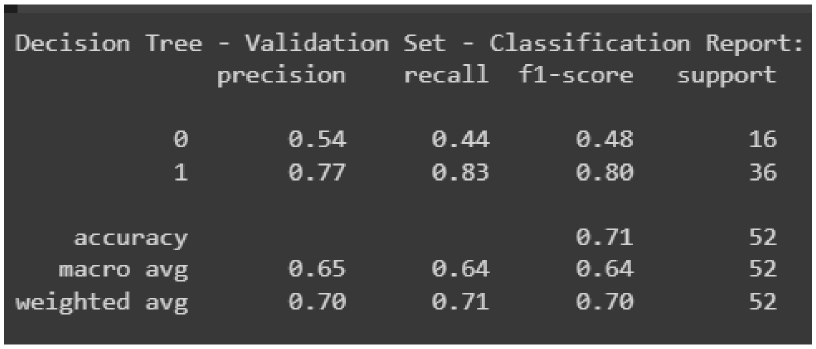


```
Decision Tree - DevSet - Accuracy: 0.7115384615384616
```

- Other performance metrics were:



```
Decision Tree - Dev Set - Confusion Matrix:
[[ 7  9]
 [ 6 30]]
```


```
Decision Tree - Validation Set - Classification Report:
              precision    recall  f1-score   support

           0       0.54      0.44      0.48        16
           1       0.77      0.83      0.80        36

    accuracy                           0.71        52
   macro avg       0.65      0.64      0.64        52
weighted avg       0.70      0.71      0.70        52
```

It can be observed that the accuracy of this model is 71%, which is the lowest among all. Hence, we do not take this model forward.

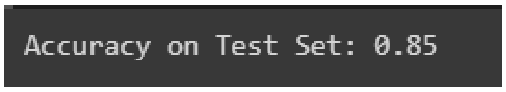
Final selected model: Linear SVM

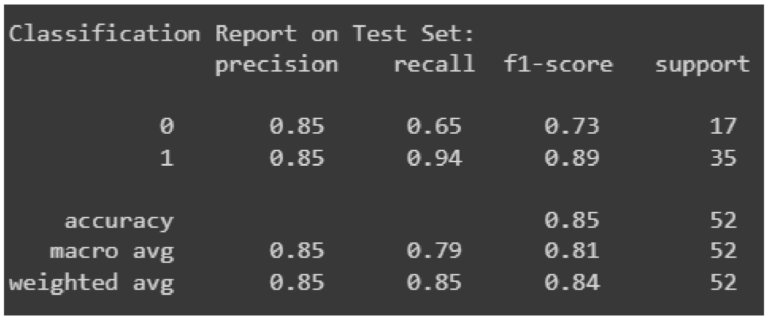
Reason for selection:

- We observed that the accuracy was almost the same for Linear SVM and logistic regression, which means that both models are well-suited for predicting the Loan Status. But if given the option to choose one - we would select Linear SVM as its accuracy is higher by some decimal points.
- Also, another reason for preferring SVM is that Linear SVM handles outliers better, as it derives a maximum margin solution.
- In addition to that, hinge loss in SVM is a better performer than log loss in Logistic Regression.

SVM Model training and evaluation-

- The best SVM model, i.e., the linear SVM, after hyperparameter tuning, was trained and tuned on the training set.
- We predicted the labels for the test set and calculated the accuracy of the prediction.


```
Accuracy on Test Set: 0.85
```


```
Classification Report on Test Set:
              precision    recall  f1-score   support

           0       0.85      0.65      0.73        17
           1       0.85      0.94      0.89        35

    accuracy                           0.85        52
   macro avg       0.85      0.79      0.81        52
weighted avg       0.85      0.85      0.84        52
```

The final accuracy of our classification model (Linear SVM) is 85%.

K-fold Cross Validation-

- We performed cross-validation on the final model for four different values of k, i.e.[3,5,7,10], and calculated mean accuracy, F1 score, precision, and recall for each of them.
- The primary reason for it was to evaluate the chosen hyperparameter value on the test data. Other reasons include assessing model stability and checking whether the model is overfitted.
- From the results, we concluded that the optimal number of folds is 10, and with the value of the hyperparameter(C=10), we will get the best accuracy.

We can also see the other evaluation metrics and conclude that the final trained model is well-learned to classify whether an individual, based on his specific features, should get a loan or not.

………………………………………………………………………………………………