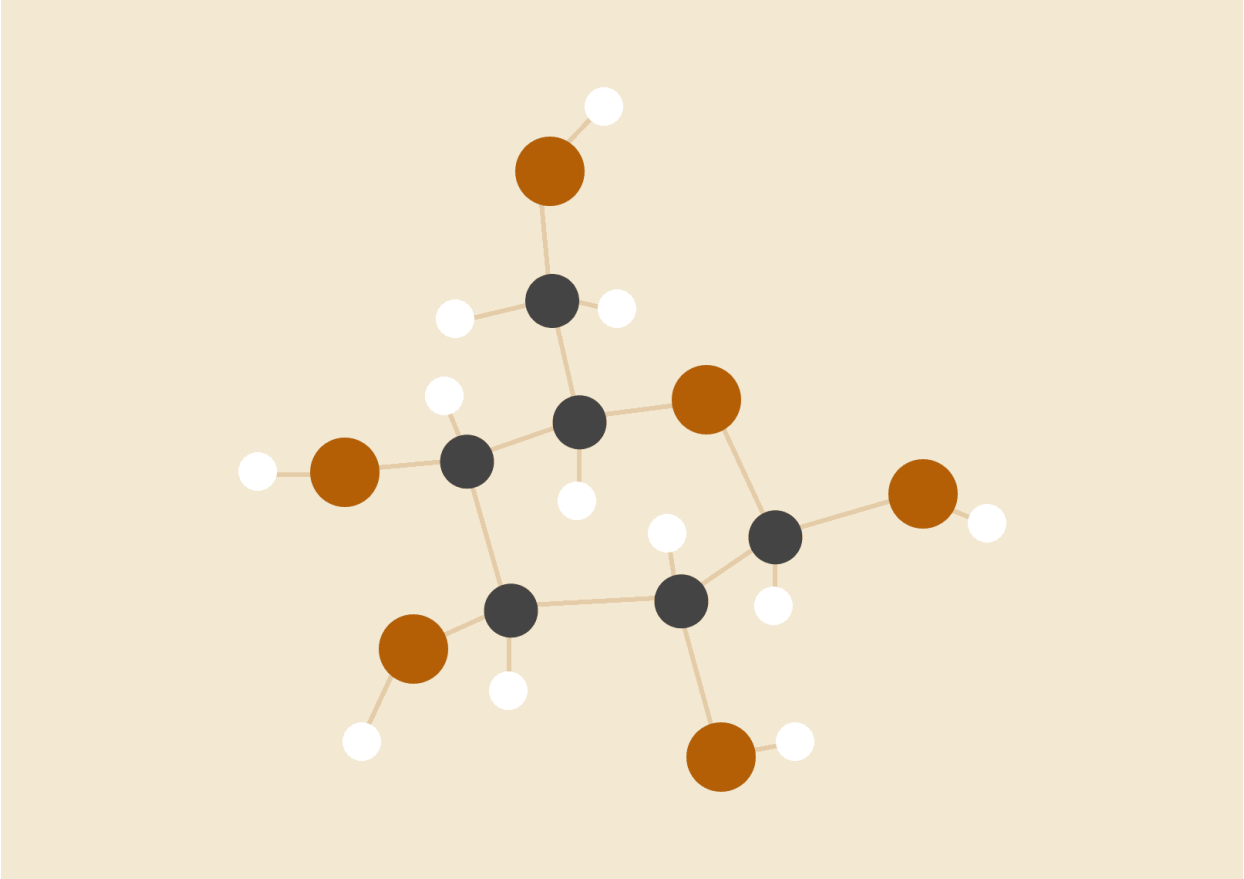
MACHINE LEARNING PROJECT REPORT

CREDIT RISK ANALYSIS USING MACHINE LEARNING



GROUP 5

RAJEEV RANJAN 20BME044
YAGNESH SHETTY 20BSM060
PULKIT SATYARTHI 20BSM043

System under study

A significant activity of the banking industry is to extend credit to customers. Credit risk management evaluates available data and decides the credibility of a customer, with the intent of protecting the financial institution against fraud.

We apply three machine learning classification algorithms. A summary of the methods are as follows:

1. Logistic Regression (LR) —Logistic Regression fits a logistic (sigmoid) function to the data. This is useful when the dependent variable is binary. To compute the weights, we will utilize the Iteratively Re-weighted Least Squares algorithm (IRLS).

2. Support Vector Machine (SVM) —SVM identifies hyperplanes to separate the data based on labels. To handle nonlinear data, projections using various kernels can be used.

3. Classification Trees –We will implement two types of classification trees: Decision Tree and Random Forest. Decision Tree is similar to a flowchart and is very easy to visualize. Every terminal node represents the output. Random Forest is a collection of decision trees, where the majority vote is taken for prediction.

Data Collection

The UC Irvine Machine Learning repository (UCI - ML) contains a collection of datasets useful for evaluating machine learning algorithms. Australian credit dataset from the UCI - ML 1 repository were used for this project.The Australian dataset has 14 characteristics and 690 instances. The response variable is a binary decision, whether a customer is credible or not. The datasets have some attributes such as the credit score, the purpose of the loan and customer information (occupation, salary, age and account duration).

Data Exploration,Feature selection and Data Cleaning

For data selection ,preprocessing,feature selection and creating dummy variable we have used various module of sklearn such as label encoders,onehotencoder and standardscaler.

```
[ ]  def data_preprocessing(dataset) :
       listt = dataset.select_dtypes(include =['category', object]).columns
       X = dataset.iloc [: , : -1].values
       y = dataset.iloc [: , -1].values
       for i in range(0 , len(y)) :
         if y[i] == 2:
           y[i] = 0

       listn = np.empty((len(listt ),1))
       for i in range (0 , len(listt )):
         listn [i ,0] = dataset.columns.get_loc(listt[i])

       # Categorial data
       from sklearn.preprocessing import LabelEncoder , OneHotEncoder
       labelencoder_X = LabelEncoder ()
       for i in range (0 , len( listn ) ):
         X[: , int(listn [i ,0]) ] = labelencoder_X .fit_transform (X[: , int (listn [i ,0]) ])
         onehotencoder = OneHotEncoder(categorical_features = [int(listn[i ,0])])

       # Splitting the dataset into the Training set and Test set
       from sklearn.model_selection import train_test_split
       X_train , X_test , y_train , y_test = train_test_split(X, y, test_size = 0.2)

       # Feature Scaling
       from sklearn.preprocessing import StandardScaler
       sc = StandardScaler ()
       X_train = sc.fit_transform(X_train)
       X_test = sc.transform(X_test)
       return X_train ,X_test , y_train , y_test ,X,y
```

Machine learning Model

LOGISTIC REGRESSION-

Suppose we have N training samples (xi , yi) where the response variable, yi , is a binary variable, i.e., yi ∈ {0, 1}. The log-odds then becomes,

$$\ln\left(\frac{p\left(y_i \mid x_i\right)}{1-p\left(y_i \mid x_i\right)}\right)=w^T x_i, \tag{3.1}$$

where p (yi | xi) is the posterior probability and w is the weight vector that we seek. Logistic Regression (LR) models are fitted via maximum likelihood. Solving eq. (3.1) for the class posterior probability,

$$\begin{aligned} Pr\left(y_i=1\right)=p\left(y_i \mid x_i\right)&=\frac{e^{w^T x_i}}{1+e^{w^T x_i}}, \\ Pr\left(y_i=0\right)=1-p\left(y_i \mid x_i\right)&=\frac{1}{1+e^{w^T x_i}}. \end{aligned} \tag{3.2}$$

We recall Bernoulli RVs (random variables) are discrete RVs with only two values, typically 0 and 1. The probability mass function is completely specified with one parameter: the “probability of success”, p = P r(X = 1),

$$pr(k)=p^k(1-p)^{(1-k)}, \quad k \in \{0,1\}.$$

We note that for binary classification, each yi can be modelled as a Bernoulli RV. Since yi is Bernoulli RV, the (joint) likelihood is

$$\mathcal{L}(w)=Pr\left(\{y_1,y_2,\ldots\} \mid w,\{x_1,x_2,\ldots\}\right)=\prod_{i=1}^{N} p^{y_i}\left(1-p\right)^{(1-y_i)}.$$

Hence,

$$
\begin{aligned}
\ln \mathcal{L}(w) &= \sum_{i=1}^{N} \left[y_i \ln(p) + (1 - y_i) \ln(1 - p) \right] \\
&= \sum_{i=1}^{N} \left[y_i \ln \left(\frac{e^{w^T x_i}}{1 + e^{w^T x_i}} \right) + (1 - y_i) \ln \left(1 - \frac{e^{w^T x_i}}{1 + e^{w^T x_i}} \right) \right] \qquad (3.3) \\
&= \sum_{i=1}^{N} \left[y_i w^T x_i - \ln \left(1 + e^{w^T x_i} \right) \right].
\end{aligned}
$$

The task is to maximize the (log) likelihood, eq. (3.3). This is equivalent to minimizing the negative log likelihood (NLL)

$$
\mathrm{NLL}(w) = - \sum_{i=1}^{N} \left[y_i w^T x_i - \ln \left(1 + e^{w^T x_i} \right) \right].
$$

Gradient descent is a first-order minimization method that uses information about the gradient. The update formula looks like

$$
w^{n+1} = w^n - \gamma \; g(w)|_{w^n} .
$$

where γ is a stepsize to be computed, and the gradient satisfies

$$
g(w) = \nabla_w \mathrm{NLL}(w) = X^T (p - y). \qquad (3.4)
$$

Here, X is N × (p + 1) matrix of xi values, p the vector of fitted probabilities has elements pi = p (xi , w), y is a vector of yi values. Newton's method is a second-order optimization method which uses the gradient and the Hessian (curvature). The update formula looks like

$$w^{n+1} = w^n - \tilde{\gamma}\, H^{-1}(w)\big|_{w^n}\ g|_{w^n}\,, \tag{3.5}$$

where ˜γ is a stepsize to be computed, the gradient is defined above in eq. (3.4), and the Hessian H satisfies,

$$H = \nabla^2 \mathrm{NLL}(w) = X^T S\, X. \tag{3.6}$$

Here, S is a N × N diagonal matrix with entries si = pi(1 − pi). Both methods converge, however, Newtons method is faster since it takes curvature into account. Substituting the definitions of H, eq. (3.6) and g eq. (3.4) into eq. (3.5), we get

$$\begin{aligned} w^{n+1} &= w^n - H^{-1}\, g \\ &= w^n - (X^T S_n\, X)^{-1} X^T\, (y - p_n) \\ &= (X^T S_n\, X)^{-1}((X^T S_n\, X)\, w^n - X^T\, (y - p_n)) \\ &= (X^T S_n\, X)^{-1}\, X^T\, (S_n\, X\, w^n + y - p_n) \\ &= (X^T S_n\, X)^{-1} X^T S_n\, z, \end{aligned} \tag{3.7}$$

where we define the working response or adjusted response,

$$z_n = X\, w^n + S_n^{-1}\, (y - p_n).$$

Iteratively Re-weighted Least Squares (IRLS)

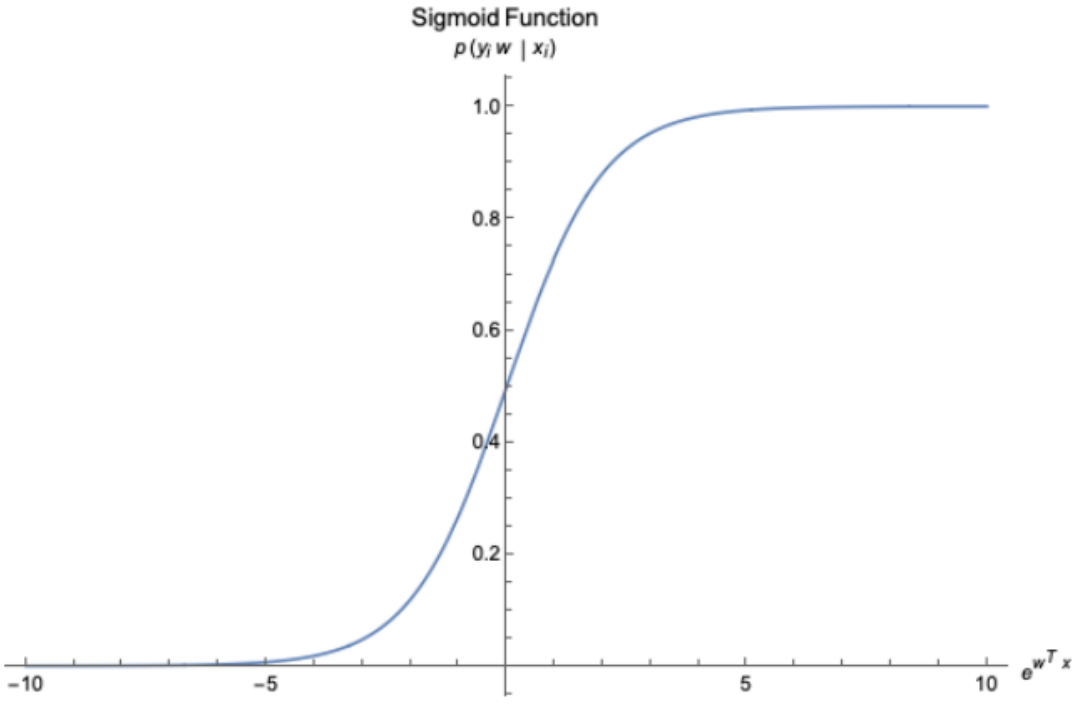
We have seen previously, eq. (3.7) that after each iteration, pn changes which in turn changes Sn and zn. Therefore we update the weights, sn, after every iteration, and solve a weighted least squares problem. The algorithm for finding the weights is summarized

in algorithm 1.

Algorithm 1 Iteratively Reweighted Least Squares (IRLS)

$w = 0$
while not converged **do**
  for $i = 1 : N$ **do**
    $p_i = \dfrac{e^{w^T x_i}}{1 + e^{w^T x_i}}$;
    $s_i = p_i(1 - p_i)$;
    $z_i = w^T x_i + \dfrac{y_i - p_i}{s_i}$
  end for
  $S = diag(s_1 s_2 \ldots s_N)$
  $w = (X^T S X)^{-1} X^T S z$
end while=0

Predictions with the Sigmoid Function: So for a new data point x, compute p(yi | w, x). If p(yi | w, x) ≥ 0.5 then classify x as Class A. If p(yi | w, x) < 0.5 then classify x as Class B.



Support Vector Machine (SVM)

Non-Linear SVM (Kernel Trick)

Suppose the data is non-linearly separable. SVM can still be applied to separate data with the use of a kernel trick – a kernel is used to transform the data to a higher dimensions which is then separable with hyperplanes. Radial Basis Functions (RBF) and polynomial kernels are two popular kernel functions used in the SVM community.

† RBF Kernel: $K(x_i, x_k) = e^{-\gamma \|x_i - x_k\|^2}$

† d-th degree Polynomial Kernel: $K(x_i, x_k) = \left(1 + x_i^T x_k\right)^d$

We can rewrite the Wolfe dual minimization problem using the kernel functions,

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k K(x_i, x_k) - \sum_{i=1}^{N} \alpha_i$$

$$\text{subject to } \sum_{i=1}^{N} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, \ldots, N$$

Decision Tree

Mathematical Framework

Suppose we view a data set in terms of feature vectors, i.e., our N × p matrix has p feature vectors,

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ \tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_p \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

The ID3 algorithm for finding the decision tree proceeds as follows.

1. Set X to be the entire data set.

2. Compute the entropy of X, where the entropy is defined as

$$H(X) = \sum_{c \in C} -p(c) \log_2 p(c).$$

For our credit risk analysis, the response variable has binary values, C = {Yes, No}, and p(c) is the respective probability.

3. Calculate the weighted sum entropy of a particular feature.

$$H(X \mid \tilde{x}) = \sum_{x} p(\tilde{x}) \sum_{c \in C} -p(c \mid \tilde{x}) \log_2 p(c \mid \tilde{x})$$

4. Calculate information gain for each feature and partition X using the maximum gain.

$$IG(X, x) = \sum_{c \in C} -p(x) \log_2 p(x) - \sum_{x} p(x) \sum_{c \in C} -p(c \mid x) \log_2 p(c \mid x)$$

5. Repeat steps 2-5 with X as each child node until we get the desired decision tree.

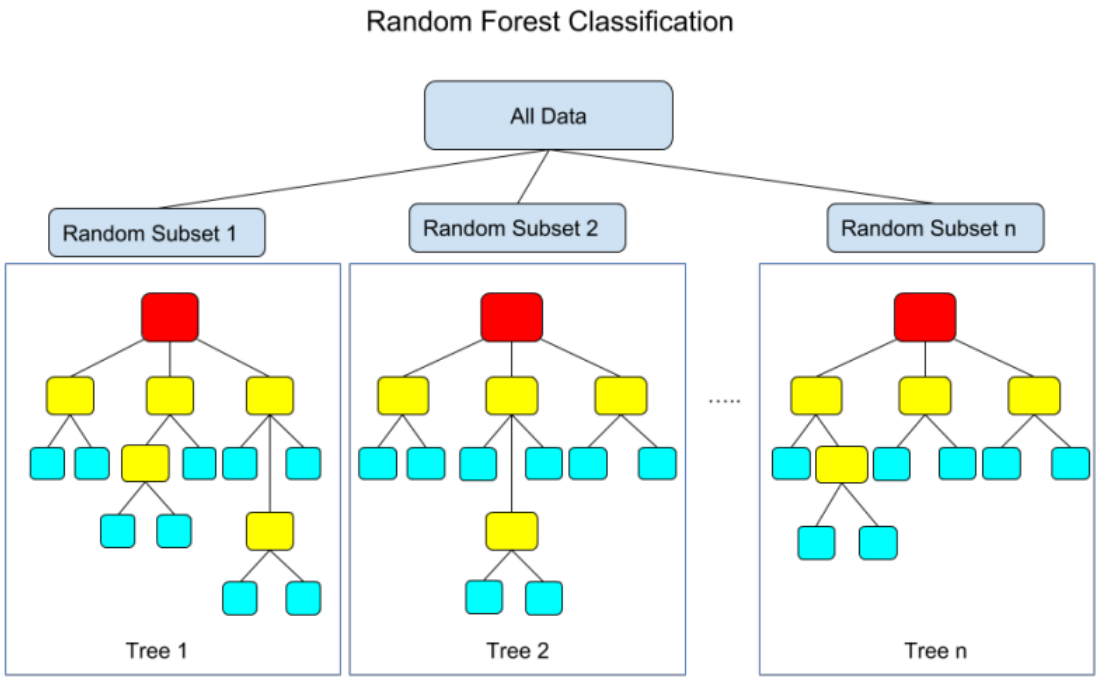
The CART algorithm is similar except that the entropy computation, eq. (6.1) is replaced by computing the gini index,

$$G(X) = \sum_{c \in C} p(c)\,(1 - p(c))$$

Random Forest Classification

Methodology

The Random Forest Classification is an algorithm which collects decision trees using the random subsets of features and choosing the majority vote among them for the classification. This method minimizes overfitting and increases the overall accuracy. Figure below shows how the random forest classification looks. So, for a given data each tree will make a vote for the prediction, and the majority vote will be considered as the prediction.



Feature Selection

In Logistic Regression

In SVM we will use parameters such as kernel , poly , rbf (radial basis function) along with various degrees sunch as 1 ,2 , 3 ,4 ,5 , 6 ,7 .

In Decision Tree the criteria used are gini and entropy along with this minimum samples split is taken into consideration with values such as 2 , 4, 6, 8, 15

In Random Forest classification criteria such as gini and entropy and minimum samples split remain same and another factor of number of samples is taken into consideration.

HyperParameters Optimisation

PARAMETRIC TUNING- it is done on various value of C and tolerance.

```
[28] # Best parameters for Logistic Regression
     parameters_lr = [{ 'C': [1 , 5 , 10] , 'tol': [1e-4 ,1e-5 ,1e-6 ,1e-10]}]
     best_para_lr = choosing_parameters ( parameters_lr , classifier_lr , X_train , y_train )
     print(best_para_lr)
     C_lr = best_para_lr [0]
     tole_lr = best_para_lr [1]

     [1, 0.0001]
```

Parametric tuning:-

The other tuning values for degree are from 1 to 7 and for kernel, RBF is also considered as different kernel. However, for Australian data set, it gives even good accuracy by setting the polynomial kernel with degree 1.

```
# Best parameters for SVM
parameters_svc = [{ 'kernel': ['rbf','poly'] , 'degree': [1 , 2 , 3 , 4, 5 , 6 , 7]}]
best_para_svc = choosing_parameters(parameters_svc , classifier_svc , X_train , y_train )
ker_svc = best_para_svc[1]
deg_svc = best_para_svc[0]
print(best_para_svc)

[1, 'rbf']
```

Parametric tuning:-

Parameters for criterion used are gini and entropy ; for mini sample split 2,4,6,8,10,15.

Out of all parameter best result comes from entropy(criterion) and 10(split).

```
[31] # Best parameters for Decision Tree
     parameters_dt = [{ 'criterion': ['gini','entropy'] , 'min_samples_split': [2 ,4 ,6 ,8 ,10 ,15]}]
     best_para_dt = choosing_parameters( parameters_dt , classifier_dt , X_train , y_train )
     criteria_dt = best_para_dt [0]
     min_split_dt = best_para_dt [1]
     print(best_para_dt)

     ['entropy', 10]
```

Parametric tuning:-

Parameters for criterion used are gini and entropy ; for mini sample split 2,5,10,15,20; for estimators (10,20,30,40,50,100,150)

Out of all parameter best result comes from gini(criterion) ,15(split) and 30(estimators).

```
# Best parameters for Random Forest
parameters_rf = [{ 'criterion': ['gini','entropy'] ,'min_samples_split': [2 ,5 , 10 , 15 , 20] ,
                  'n_estimators': [10 , 20 , 30 , 40 , 50 ,100 ,150]}]
best_para_rf = choosing_parameters ( parameters_rf ,classifier_rf , X_train , y_train )
print(best_para_rf)
min_samp_spl_rf = best_para_rf [1]
n_est_rf = best_para_rf [2]
criterion_rf = best_para_rf [0]

['gini', 15, 30]
```

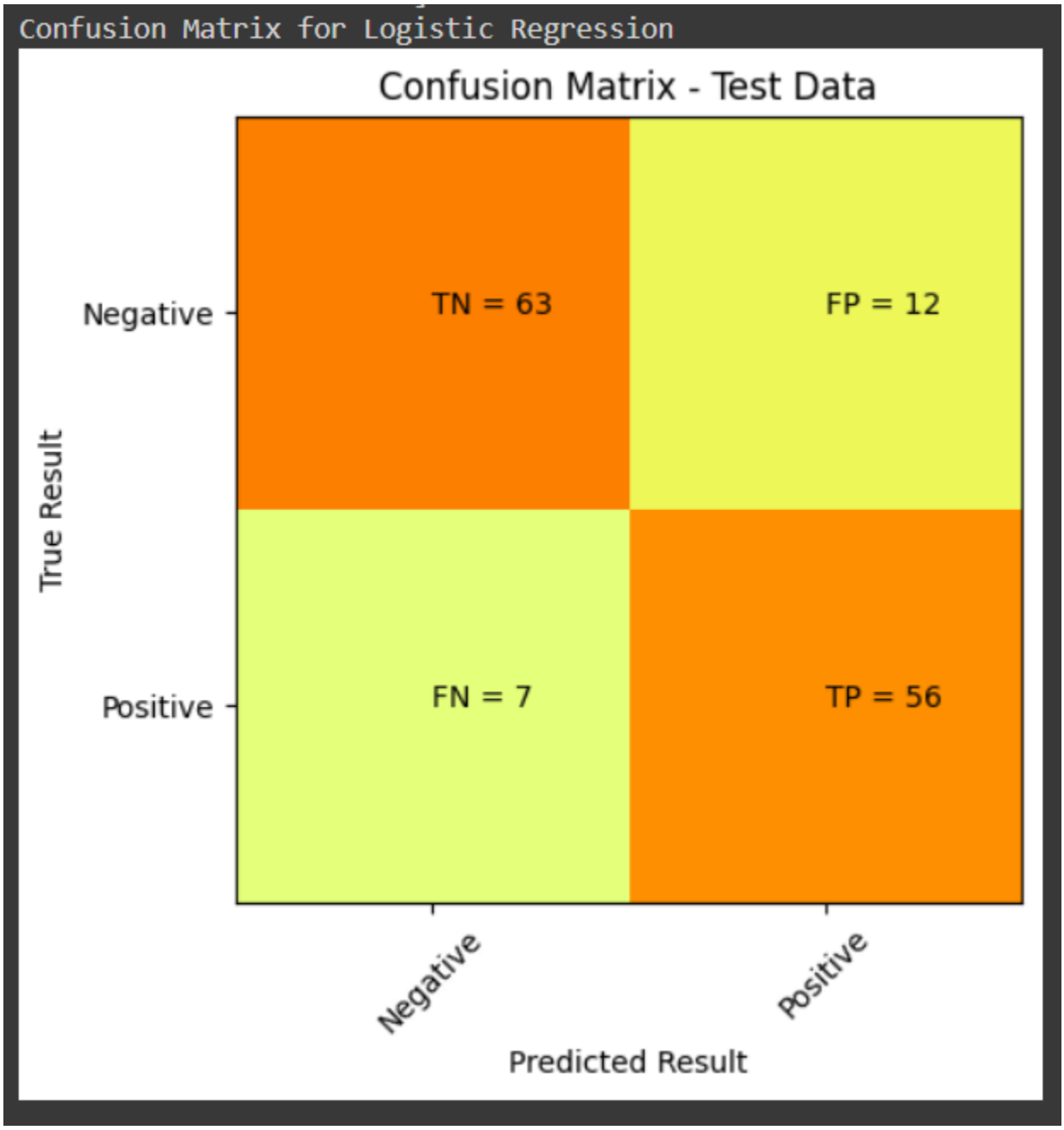
Coding the Actual System

The code of the system can be found in the attachment with the report.

Result and Analysis

Mean accuracy and confusion matrix before tuning in logistic Regression are as follows-:

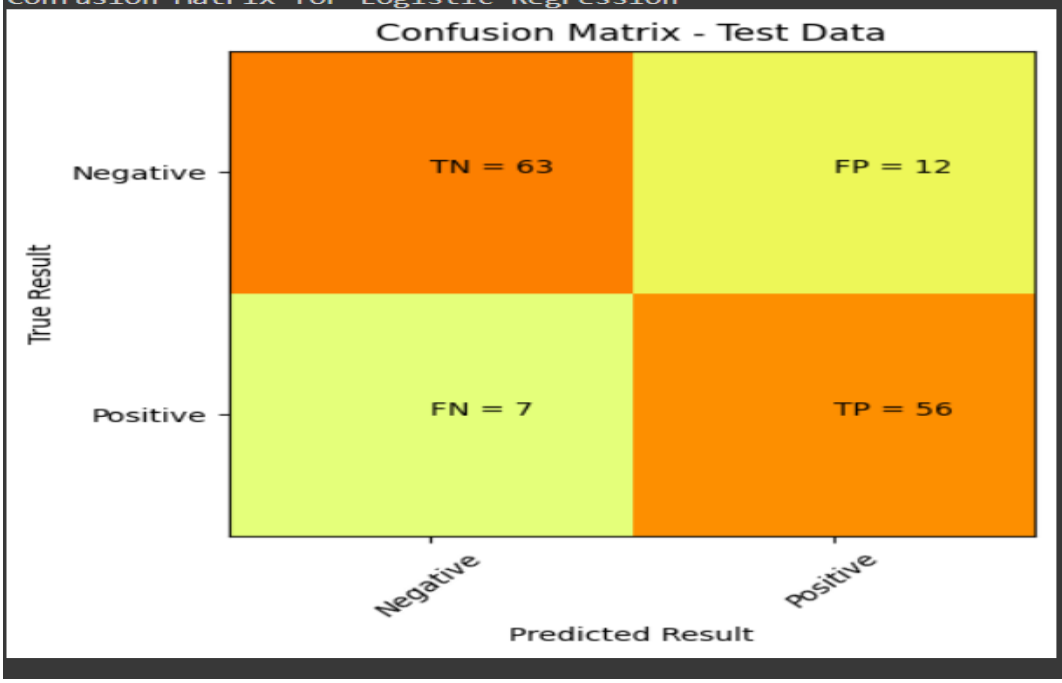
```
LR: Mean Accuracy before tuning  86.21428571428572
```

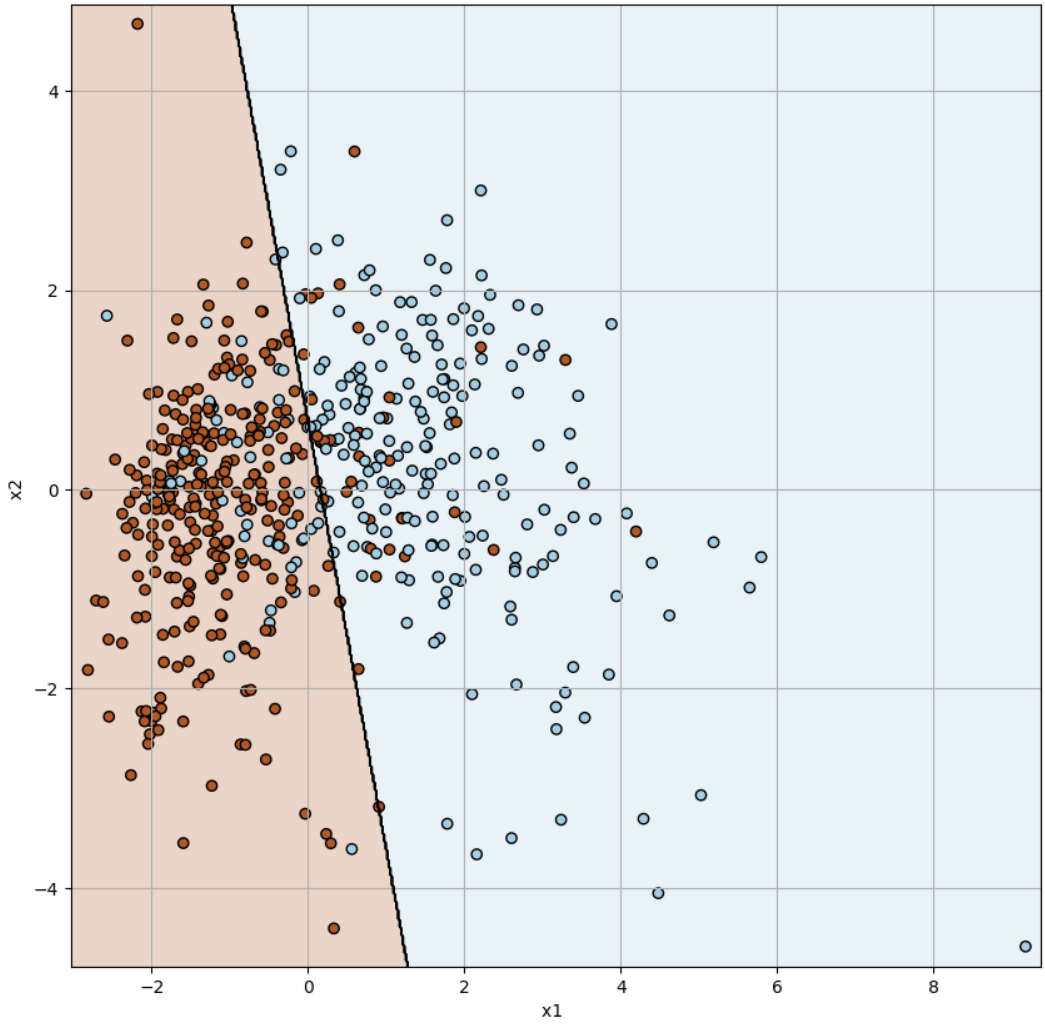


Mean accuracy and confusion matrix after tuning are as follows-:

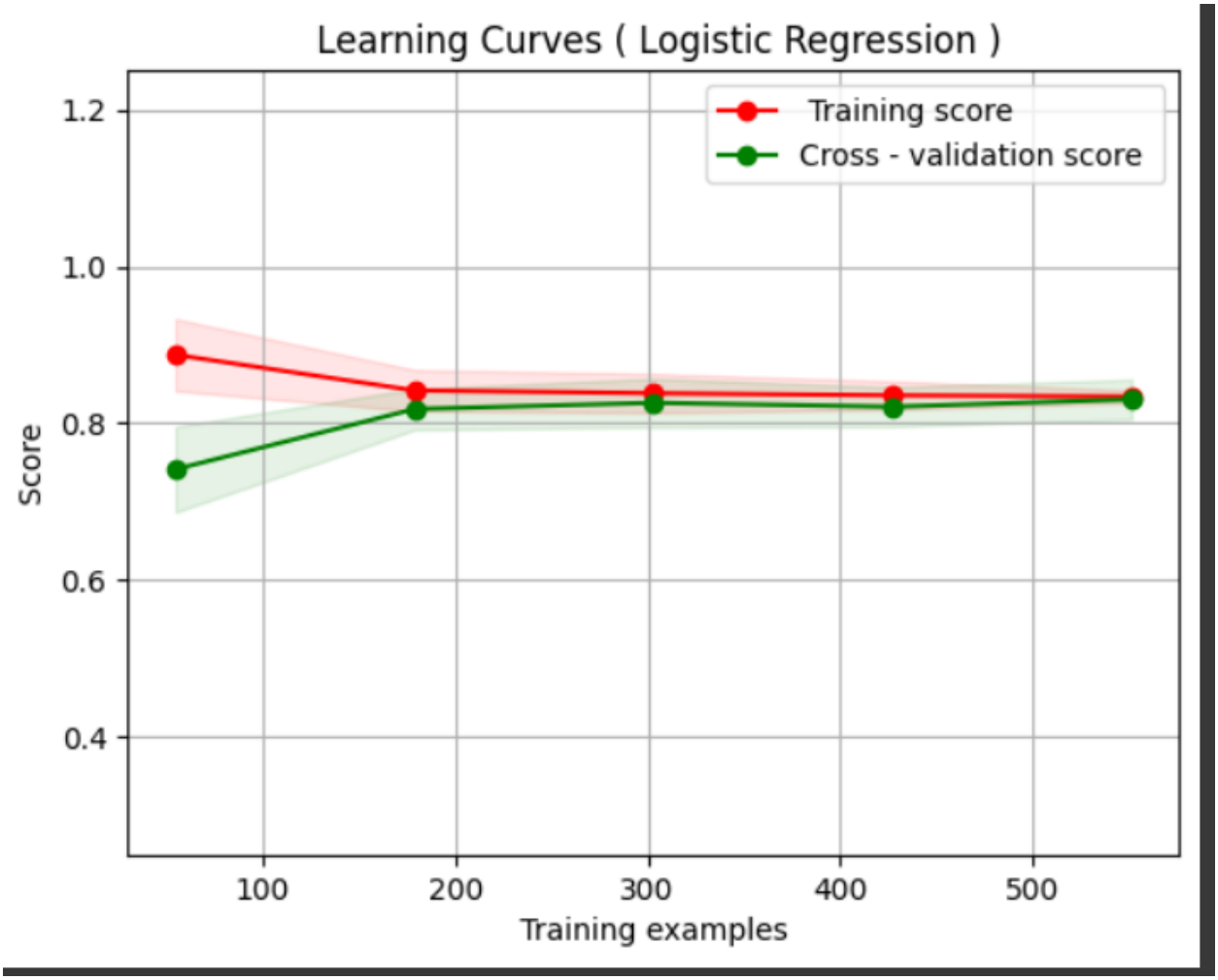
```
LR : Mean Accuract after tuning  86.21428571428572
```

```
Confusion Matrix for Logistic Regression
```





Decision boundary plot using Logistic Regression of the Australian data set.



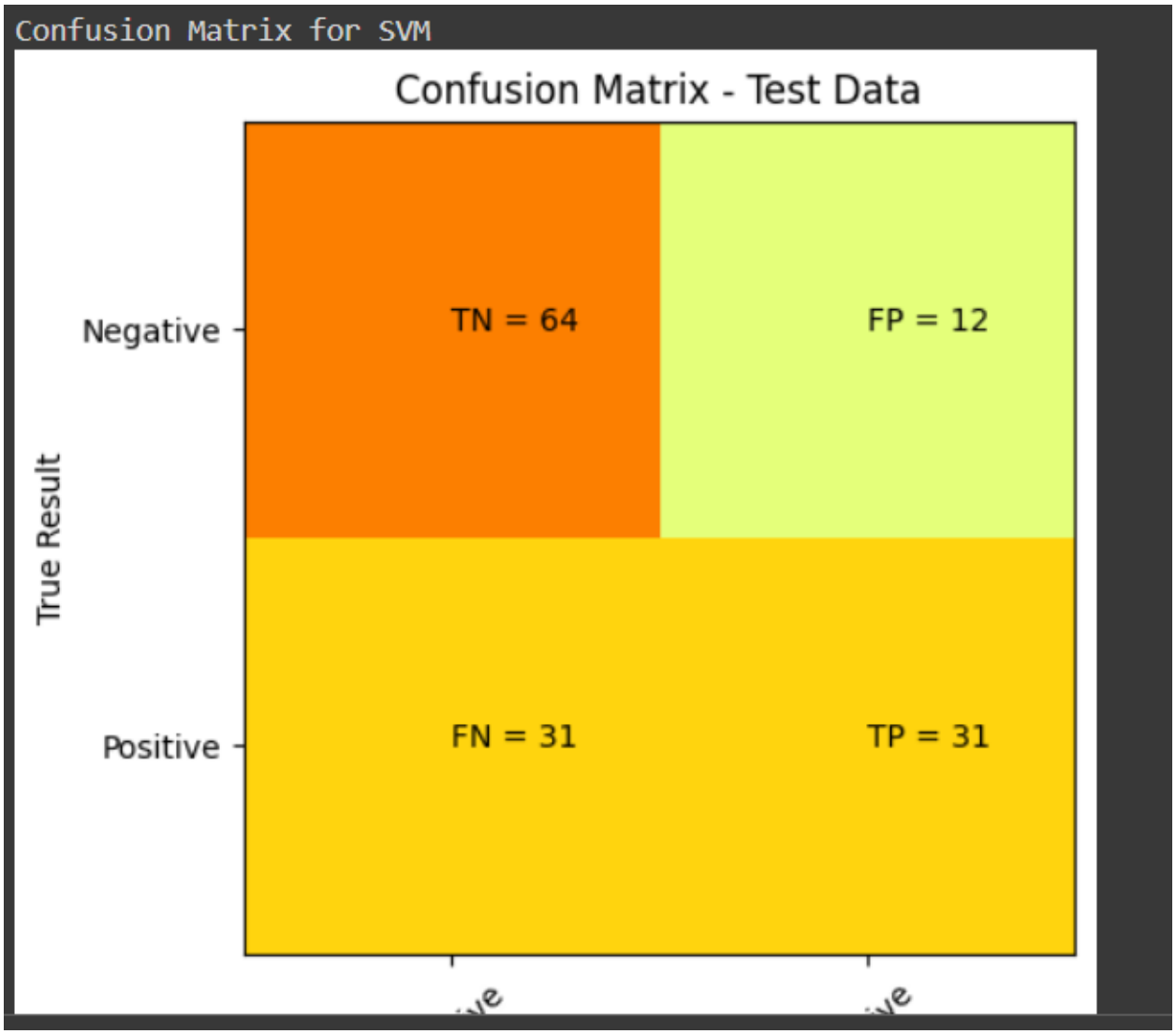
Learning curve of Logistic Regression using the Australian data set. No high bias or high variance is observed, indicating that the LR model is not over-fitting or under-fitting this Australian data set.

Results (both default and after tuning)

By applying Kernel SVM to Australian credit data sets, the accuracies are obtained shown in table.The default parameters for SVM is set to be polynomial kernel with degree 2.

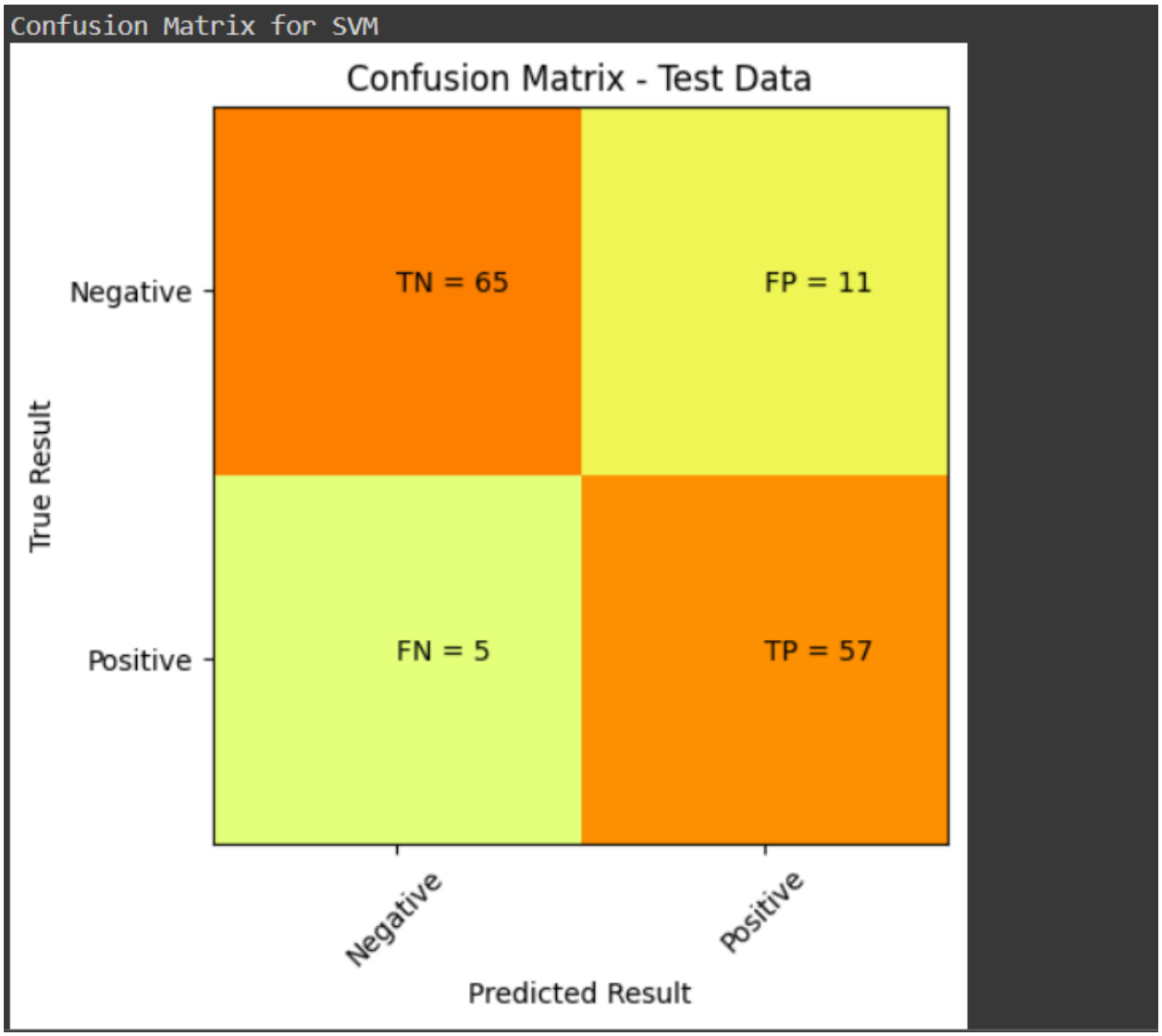
Mean accuracy and confusion matrix before tuning are as follows-:

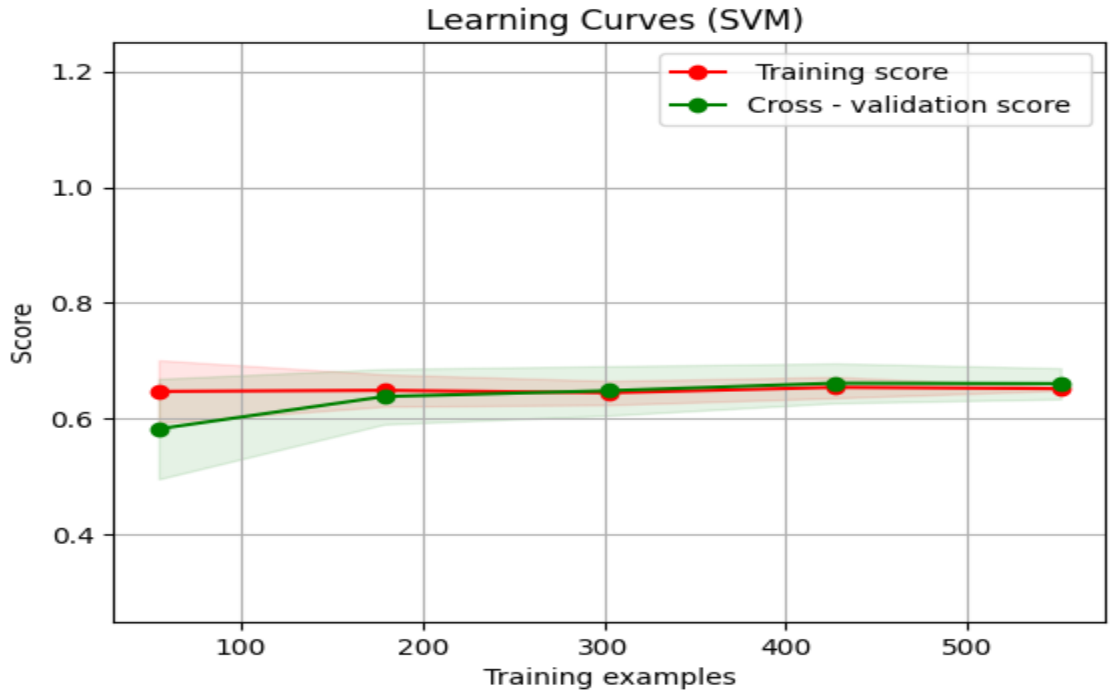
```
SVC : Mean Accuracy before tuning  71.14285714285715
```



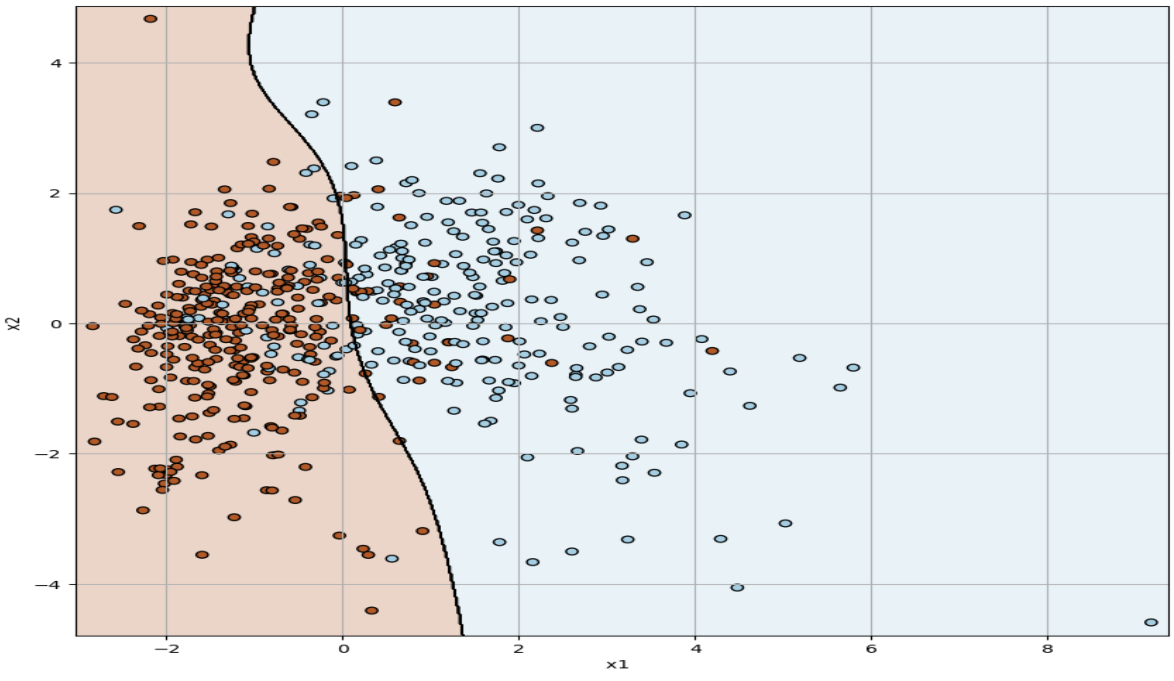
Mean accuracy and confusion matrix after tuning are as follows-:

```
SVM : Mean Accuracy after tuning  85.30194805194805
```


Confusion Matrix for SVM
Confusion Matrix - Test Data
TN = 65
FP = 11
FN = 5
TP = 57
Negative
Positive
True Result
Negative
Positive
Predicted Result



Learning curve of Support Vector Machine using the Australian data set. No gap is observed as increases the samples, indicating that the Support Vector Machine model is under-fitting this Australian data set.

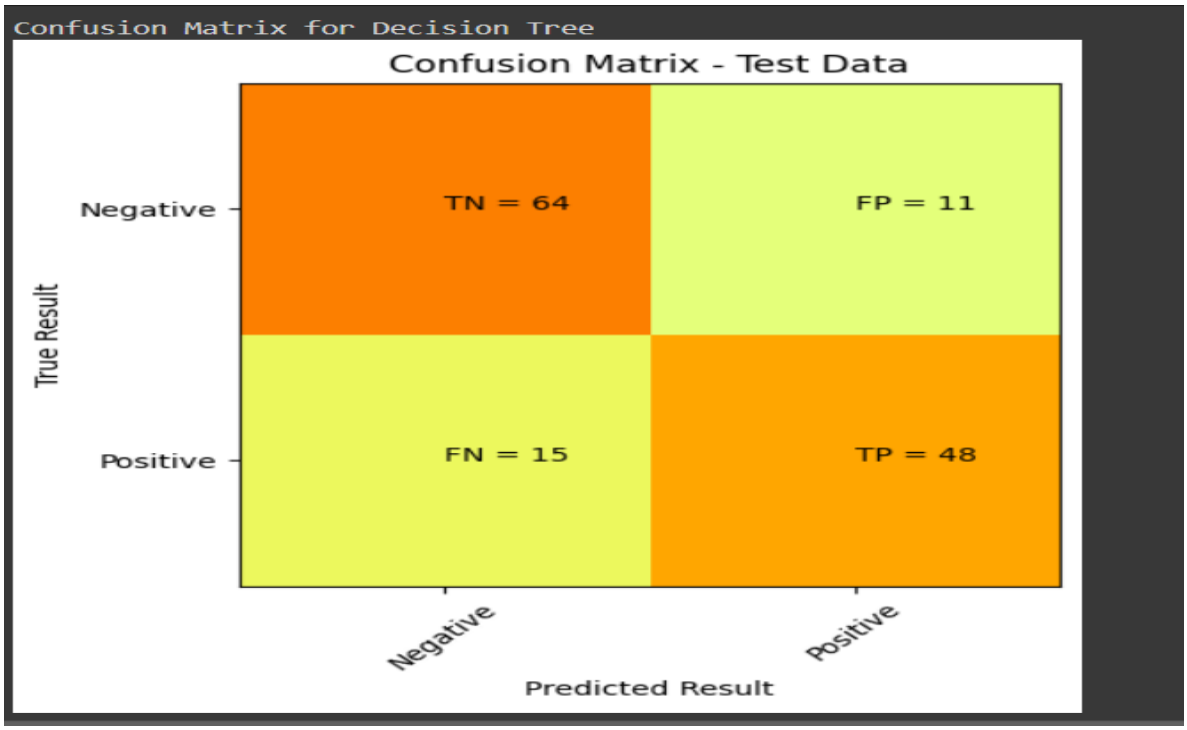


Decision boundary plot using Support Vector Machine of Australian data set.

Results(both default and after tuning)-

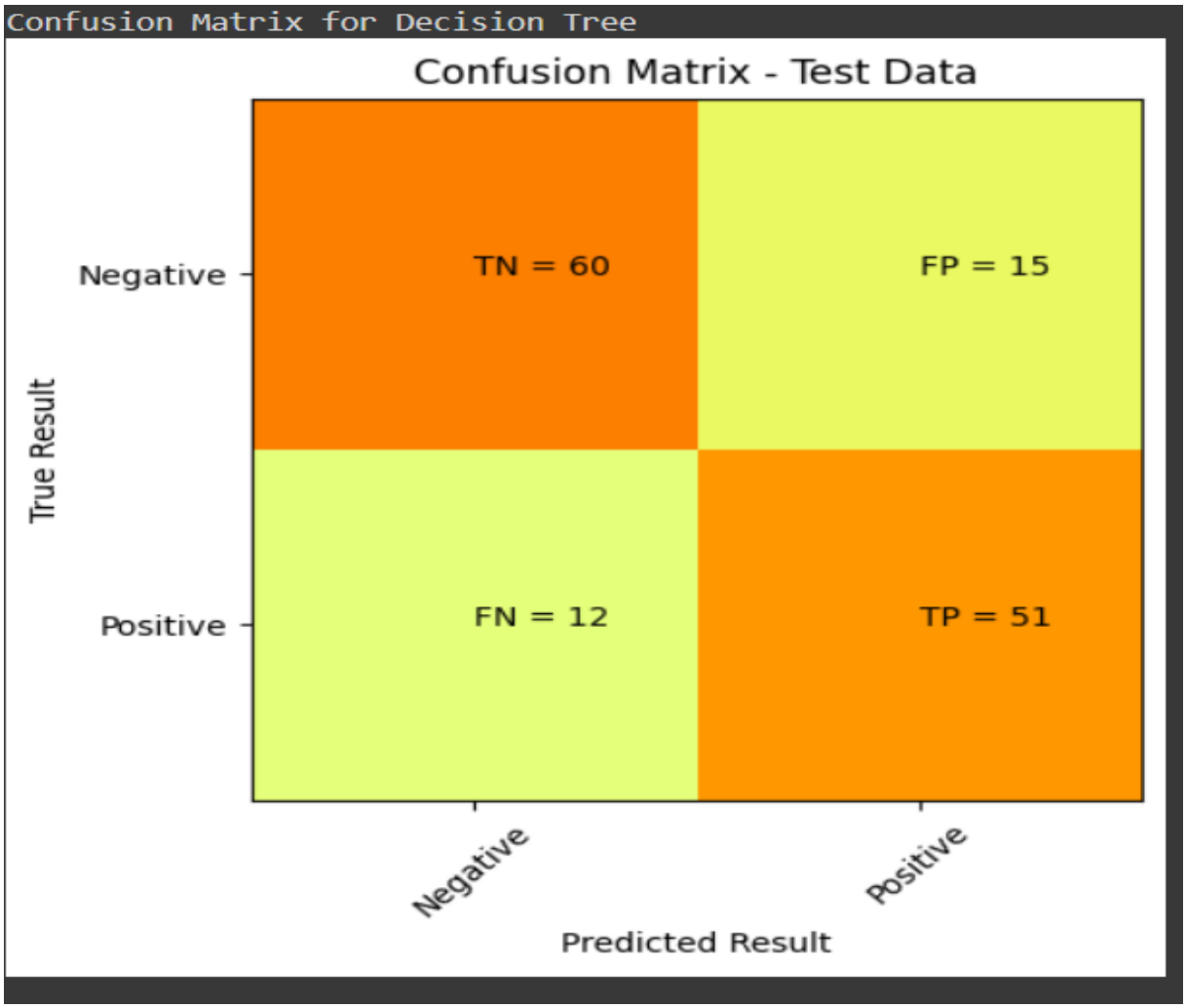
Mean accuracy and confusion matrix without tuning:-

```
Decision Tree : Mean Accuracy before tuning  84.57792207792207
```

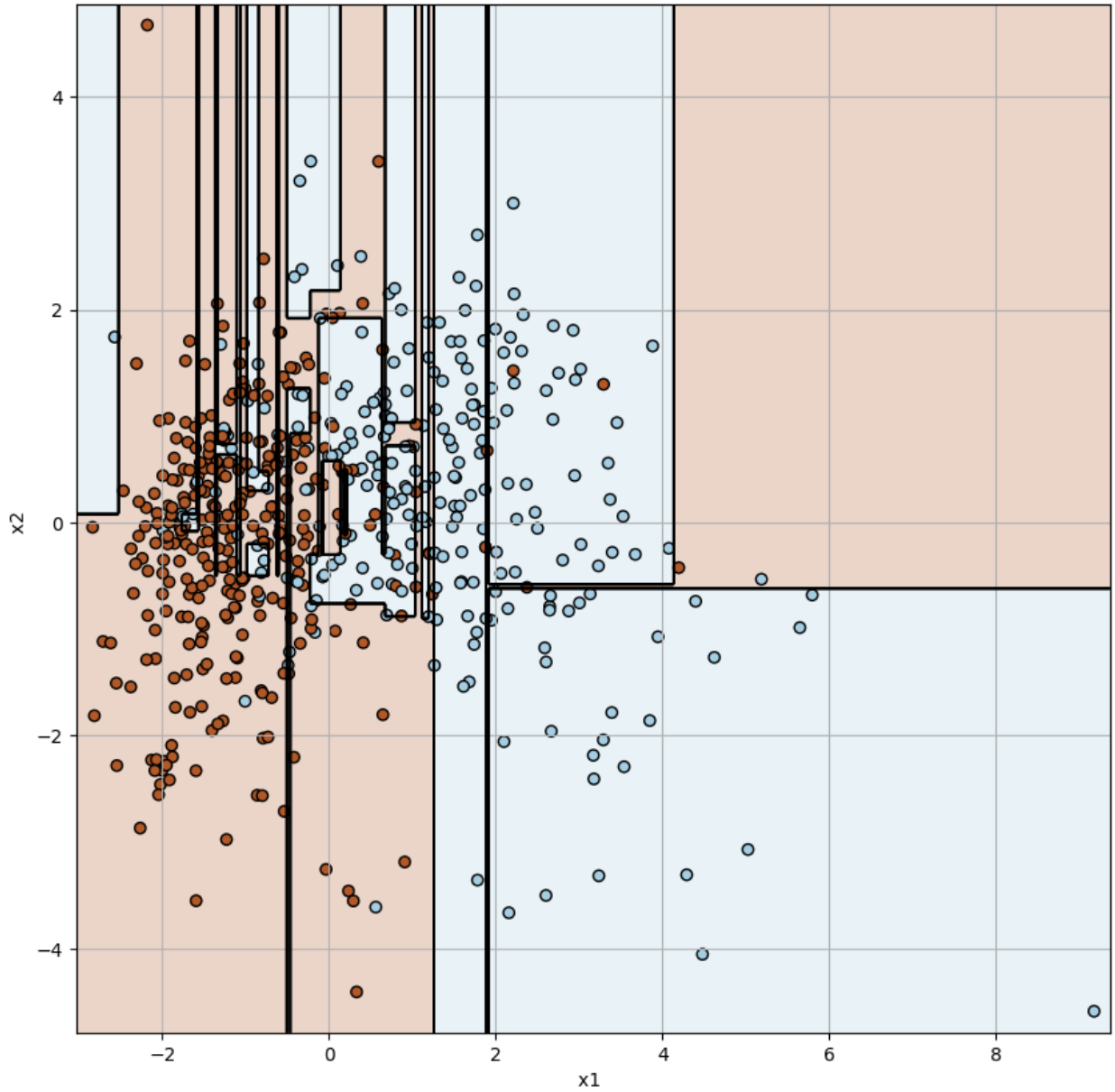


Mean accuracy and confusion matrix with tuning:-

```
Decision Tree : Mean Accuracy after tuning  85.47727272727272
```


Confusion Matrix for Decision Tree
Confusion Matrix - Test Data
Negative
Positive
True Result
TN = 60
FP = 15
FN = 12
TP = 51
Negative
Positive
Predicted Result


Learning Curves (Decision Tree)
Training score
Cross - validation score
1.2
1.0
0.8
0.6
0.4
Score
100
200
300
400
500
Training examples



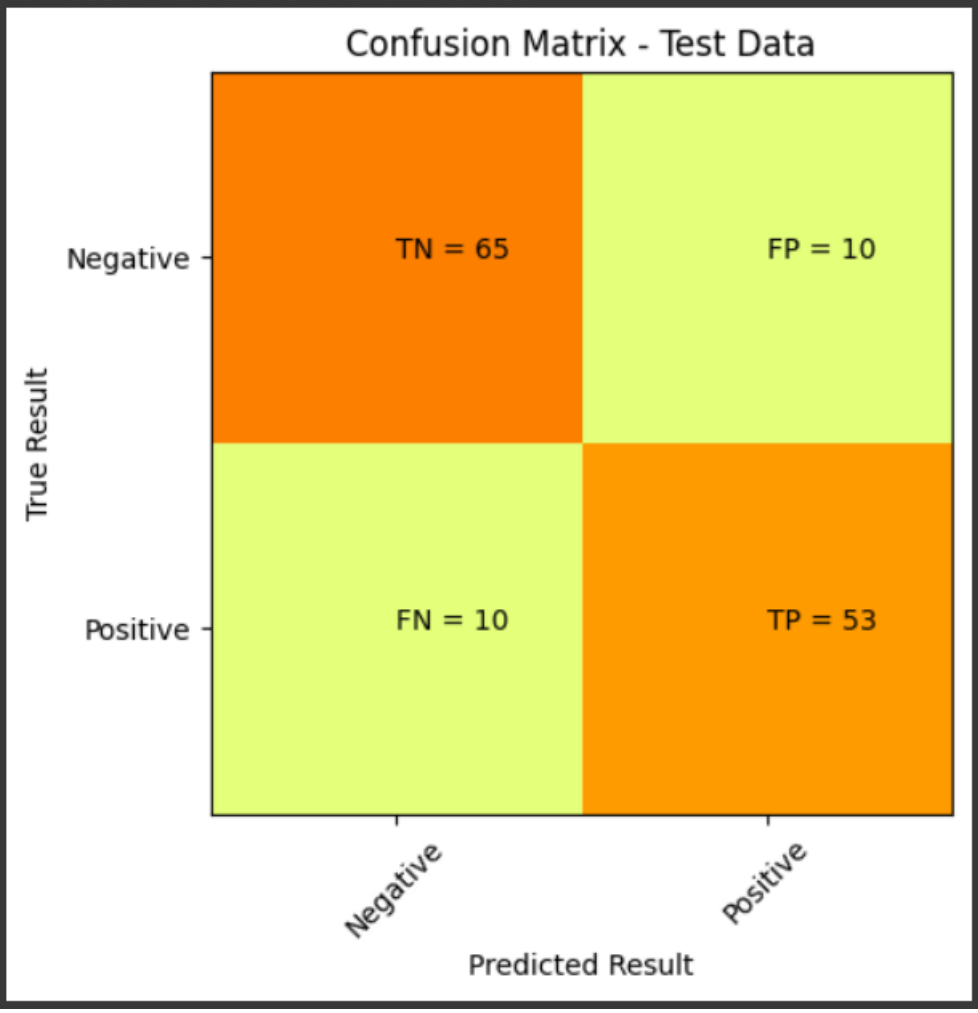
Decision boundary plot using Decision Tree of Australian data set.

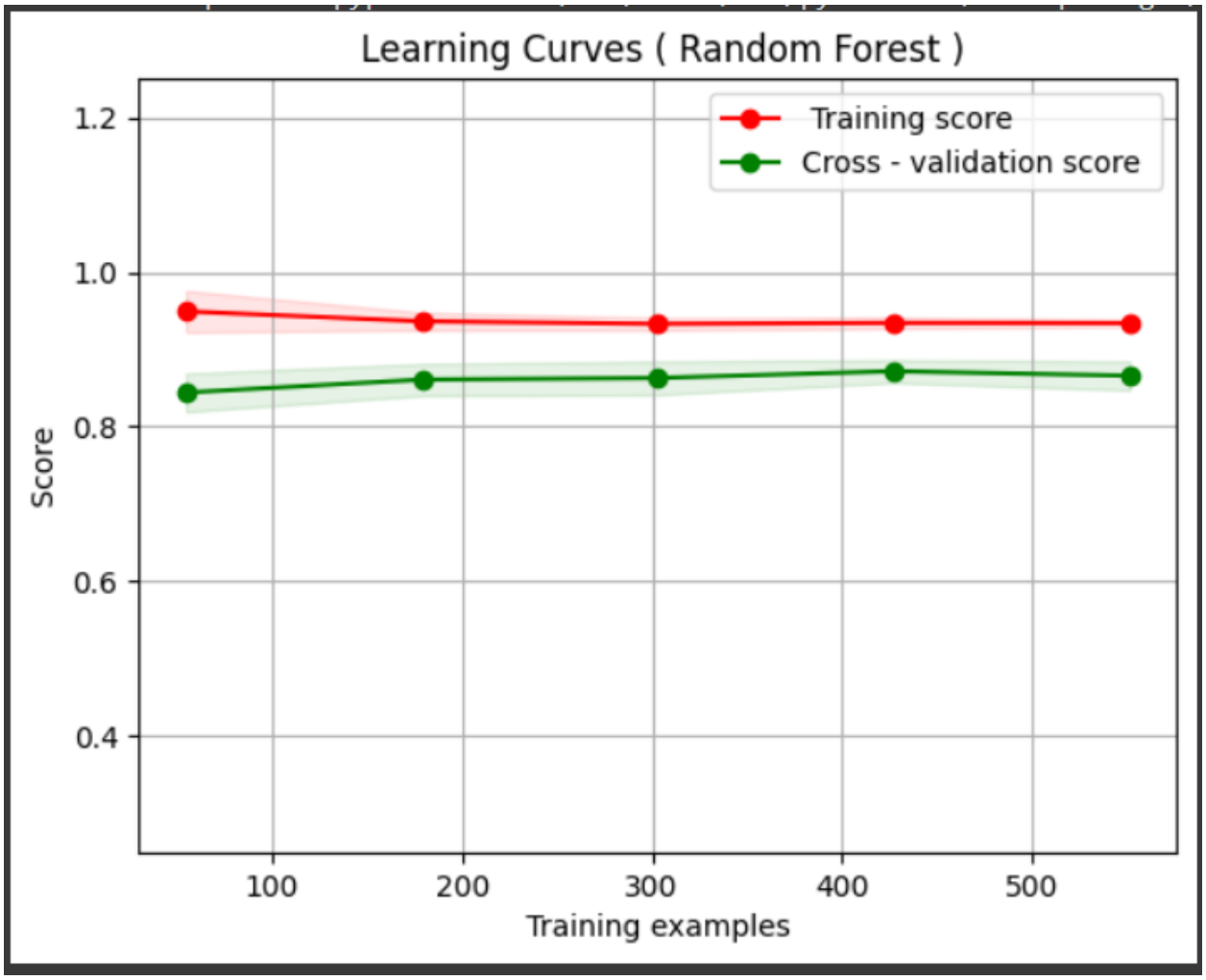
Results(both default and after tuning)-

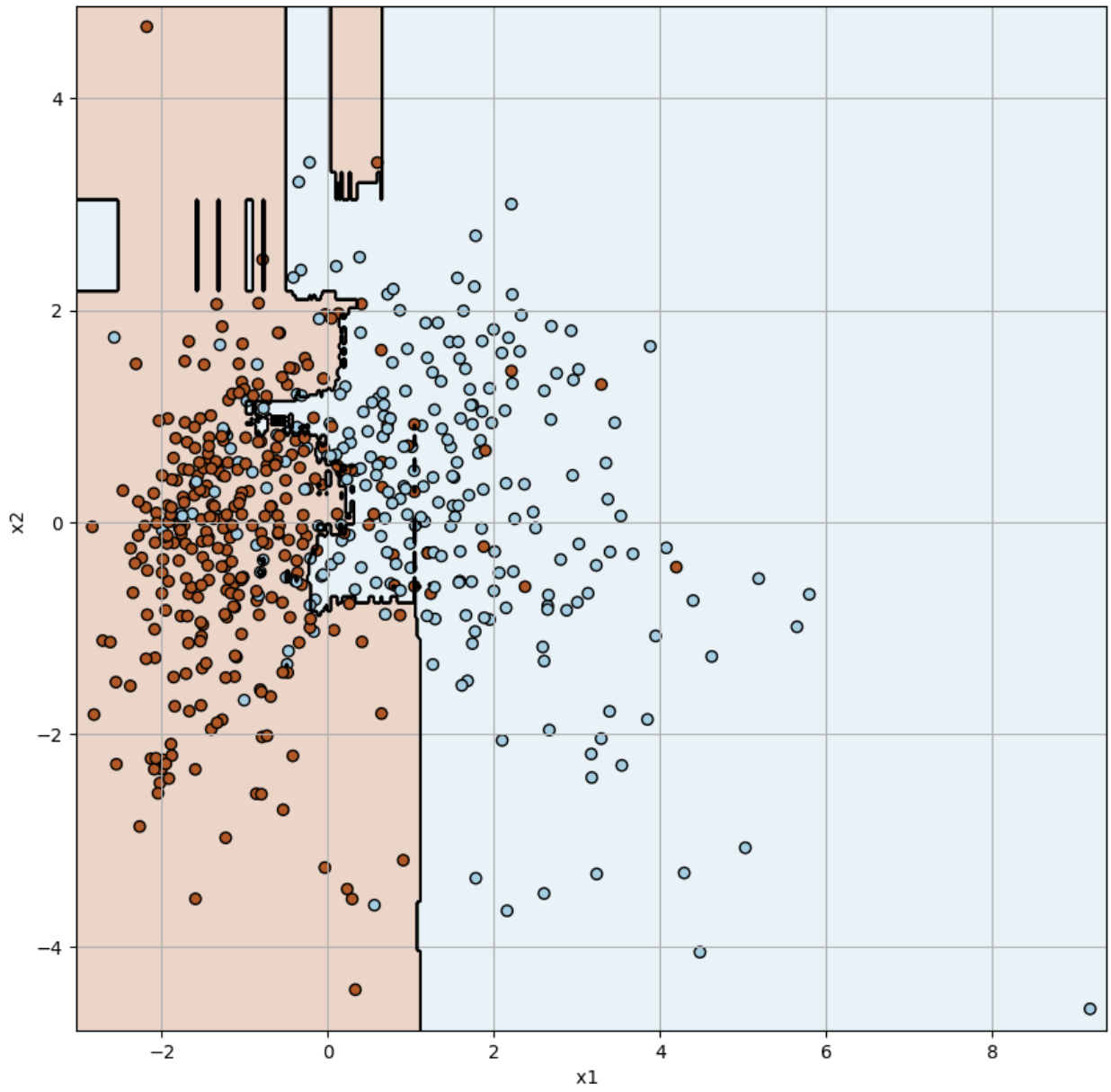
Mean accuracy and confusion matrix without tuning:-

```
Random Forest : Mean Accuracy before tuning  85.2987012987013
```

Confusion Matrix for Random Forest




Learning Curves (Random Forest)
Training score
Cross - validation score
1.2
1.0
0.8
0.6
0.4
Score
100
200
300
400
500
Training examples



After observing the results we come to the conclusion that Classification Trees Provide best accuracy to the given dataset and avoid overfitting which is later accompanied by Logistic Regression and atlast SVM comes into the picture.

By doing the Parameter Optimization we have increased the accuracy of SVM manifold but the accuracy of other models have changed a bit.