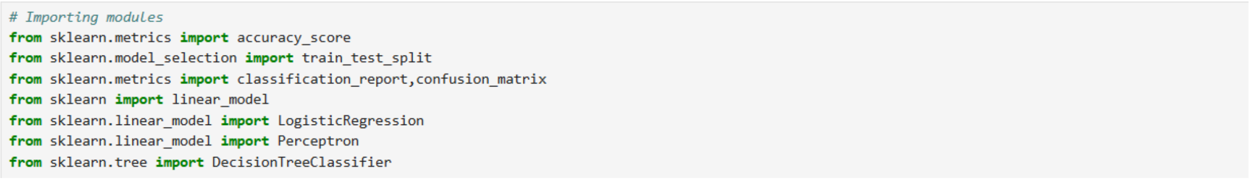
# Model Development Phase

| | |
|---|---|
| Date | 4th June 2024 |
| Team ID | SWTID1720175375 |
| Project Title | Prediction and analysis of liver patient data using ML |
| Maximum Marks | 4 Marks |

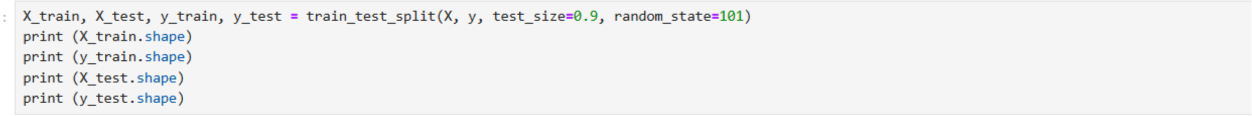
## Initial Model Training Code, Model Validation and Evaluation Report

## Initial Model Training Code:



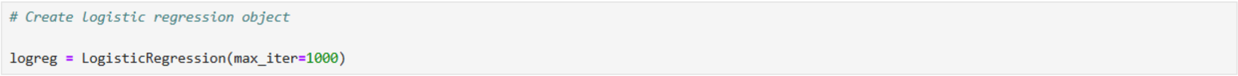
```
# Importing modules
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Perceptron
from sklearn.tree import DecisionTreeClassifier
```

Train Test Split:

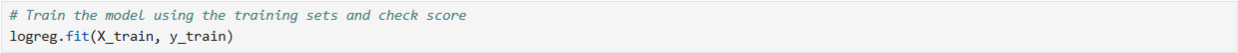


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.9, random_state=101)
print (X_train.shape)
print (y_train.shape)
print (X_test.shape)
print (y_test.shape)
```

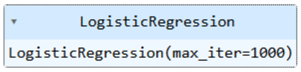
## Logistic Regression:



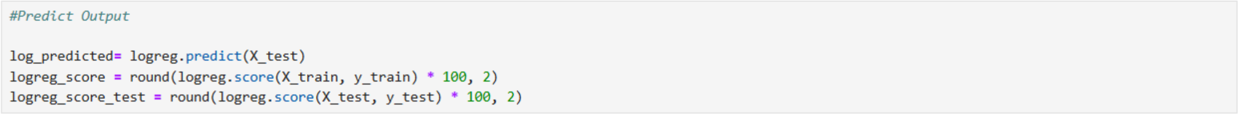
```
# Create logistic regression object

logreg = LogisticRegression(max_iter=1000)
```



```
# Train the model using the training sets and check score
logreg.fit(X_train, y_train)
```

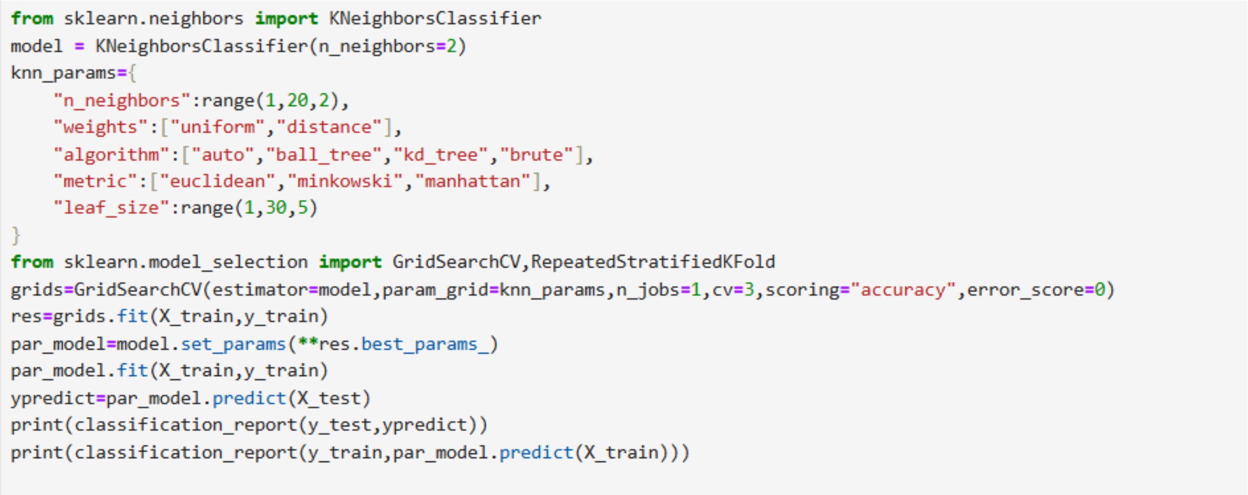


```
LogisticRegression
LogisticRegression(max_iter=1000)
```



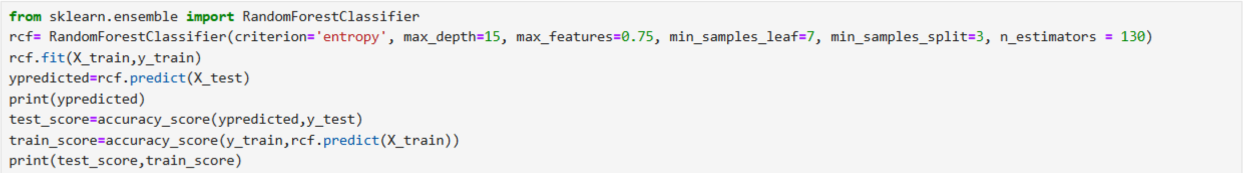
```
#Predict Output

log_predicted= logreg.predict(X_test)
logreg_score = round(logreg.score(X_train, y_train) * 100, 2)
logreg_score_test = round(logreg.score(X_test, y_test) * 100, 2)
```

## KNN:



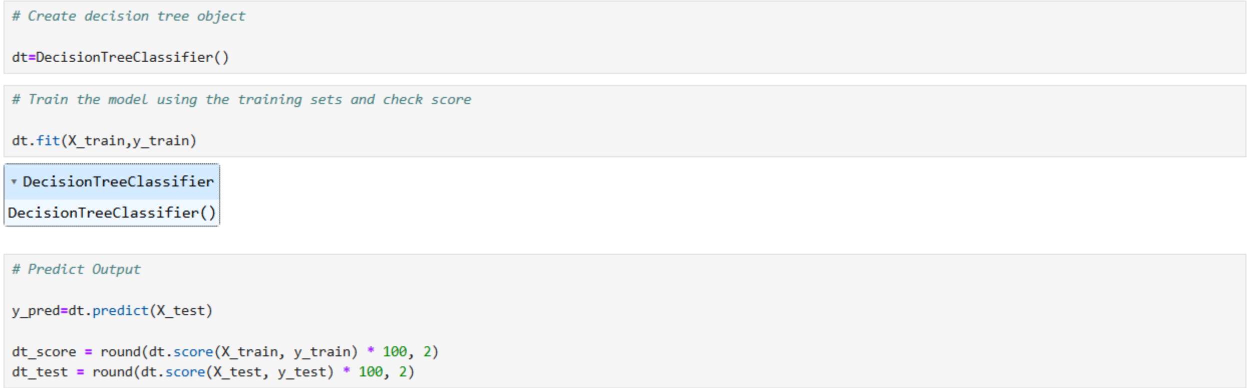
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=2)
knn_params={
    "n_neighbors":range(1,20,2),
    "weights":["uniform","distance"],
    "algorithm":["auto","ball_tree","kd_tree","brute"],
    "metric":["euclidean","minkowski","manhattan"],
    "leaf_size":range(1,30,5)
}
from sklearn.model_selection import GridSearchCV,RepeatedStratifiedKFold
grids=GridSearchCV(estimator=model,param_grid=knn_params,n_jobs=1,cv=3,scoring="accuracy",error_score=0)
res=grids.fit(X_train,y_train)
par_model=model.set_params(**res.best_params_)
par_model.fit(X_train,y_train)
ypredict=par_model.predict(X_test)
print(classification_report(y_test,ypredict))
print(classification_report(y_train,par_model.predict(X_train)))
```

## Random Forest classifier:



```
from sklearn.ensemble import RandomForestClassifier
rcf= RandomForestClassifier(criterion='entropy', max_depth=15, max_features=0.75, min_samples_leaf=7, min_samples_split=3, n_estimators = 130)
rcf.fit(X_train,y_train)
ypredicted=rcf.predict(X_test)
print(ypredicted)
test_score=accuracy_score(ypredicted,y_test)
train_score=accuracy_score(y_train,rcf.predict(X_train))
print(test_score,train_score)
```

## Decision tree classifier:



```
# Create decision tree object

dt=DecisionTreeClassifier()
```

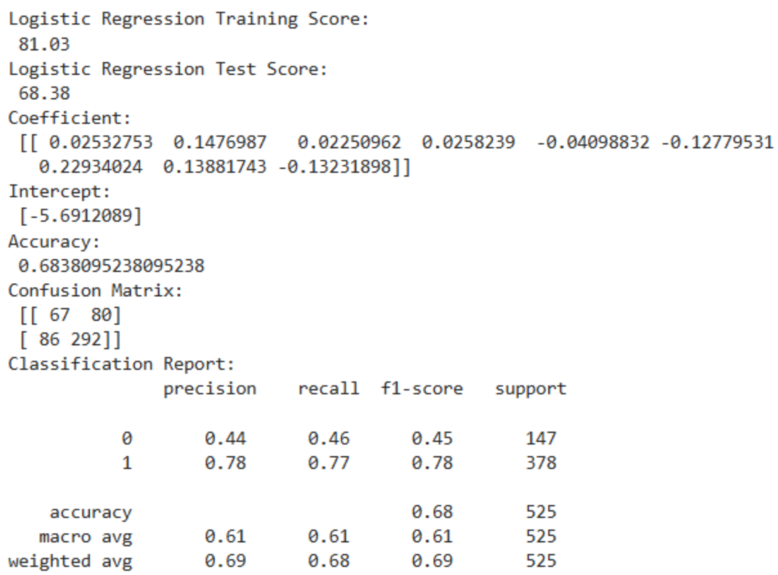
```
# Train the model using the training sets and check score

dt.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
# Predict Output

y_pred=dt.predict(X_test)

dt_score = round(dt.score(X_train, y_train) * 100, 2)
dt_test = round(dt.score(X_test, y_test) * 100, 2)
```

# Model Validation and Evaluation Report:

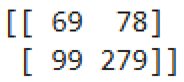
## Model 1(Logistic Regression):



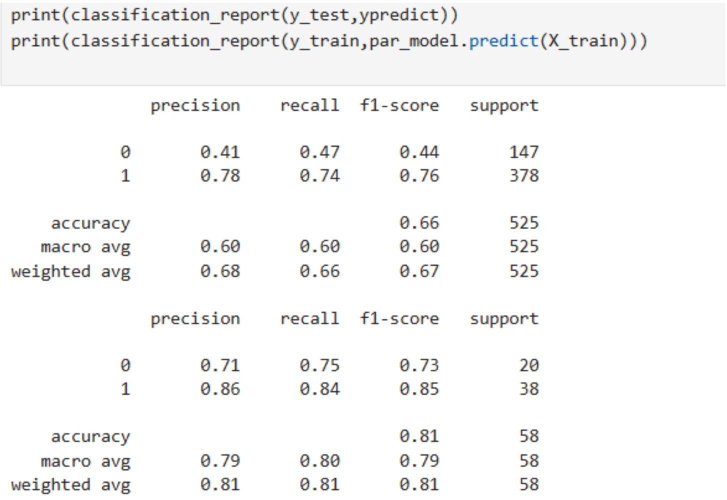
```
Logistic Regression Training Score:
 81.03
Logistic Regression Test Score:
 68.38
Coefficient:
 [[ 0.02532753  0.1476987   0.02250962  0.0258239  -0.04098832 -0.12779531
   0.22934024  0.13881743 -0.13231898]]
Intercept:
 [-5.6912089]
Accuracy:
 0.6838095238095238
Confusion Matrix:
 [[ 67  80]
 [ 86 292]]
Classification Report:
               precision    recall  f1-score   support

           0       0.44      0.46      0.45       147
           1       0.78      0.77      0.78       378

    accuracy                           0.68       525
   macro avg       0.61      0.61      0.61       525
weighted avg       0.69      0.68      0.69       525
```

## Model-2(KNN):

## Confusion matrix:



```
[[ 69  78]
 [ 99 279]]
```

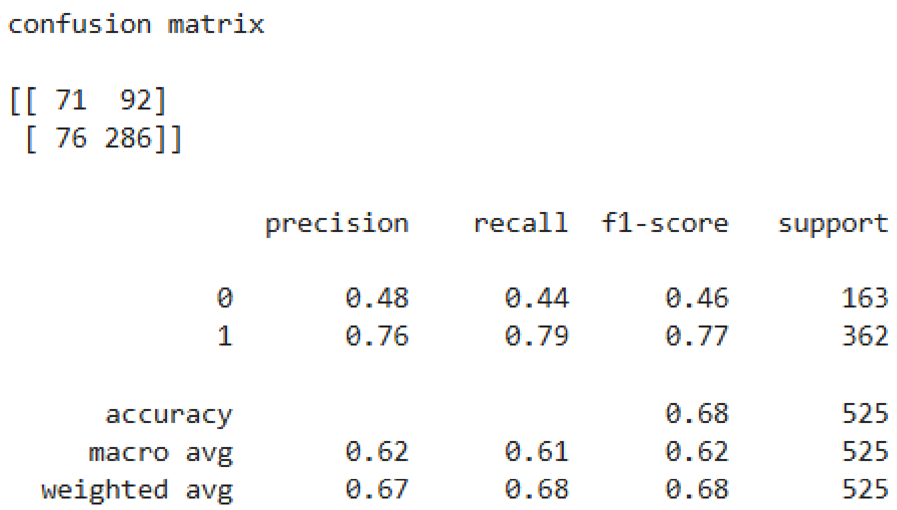


```
print(classification_report(y_test,ypredict))
print(classification_report(y_train,par_model.predict(X_train)))
```

```
              precision    recall  f1-score   support

           0       0.41      0.47      0.44       147
           1       0.78      0.74      0.76       378

    accuracy                           0.66       525
   macro avg       0.60      0.60      0.60       525
weighted avg       0.68      0.66      0.67       525

              precision    recall  f1-score   support

           0       0.71      0.75      0.73        20
           1       0.86      0.84      0.85        38

    accuracy                           0.81        58
   macro avg       0.79      0.80      0.79        58
weighted avg       0.81      0.81      0.81        58
```

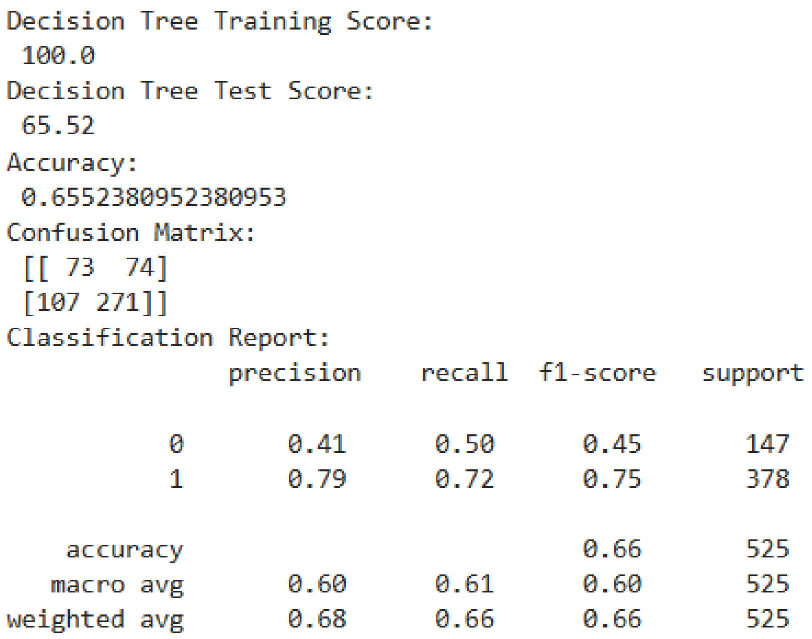
## Model-3( Random forest Classifier):

### Test Data:



```
 confusion matrix

[[ 71  92]
 [ 76 286]]

              precision    recall  f1-score   support

           0       0.48      0.44      0.46       163
           1       0.76      0.79      0.77       362

    accuracy                           0.68       525
   macro avg       0.62      0.61      0.62       525
weighted avg       0.67      0.68      0.68       525
```

## Model-4(Decision Tree Classifier):



```
Decision Tree Training Score:
 100.0
Decision Tree Test Score:
 65.52
Accuracy:
 0.6552380952380953
Confusion Matrix:
 [[ 73  74]
 [107 271]]
Classification Report:
              precision    recall  f1-score   support

           0       0.41      0.50      0.45       147
           1       0.79      0.72      0.75       378

    accuracy                           0.66       525
   macro avg       0.60      0.61      0.60       525
weighted avg       0.68      0.66      0.66       525
```