Team ID	PNT2022TMID52707
Project Name	Project - Statistical Machine Learning Approaches to Liver Disease Prediction.

Train And Test The Model Using Classification Algorithms

```
x=data.drop(columns='Dataset',axis=1)
y=data['Dataset']
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,stratify=y,random_state=42)
```

```
print(x.shape,x_train.shape,x_test.shape)
```

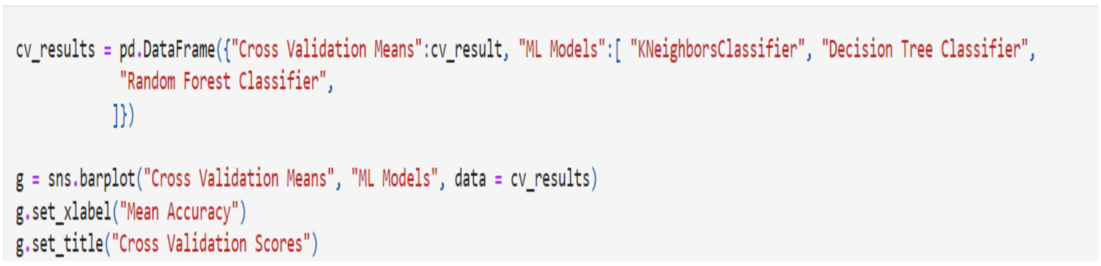
```
(1636, 10) (1145, 10) (491, 10)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
xtrain=sc.fit_transform(x_train)
xtest=sc.transform(x_test)
```

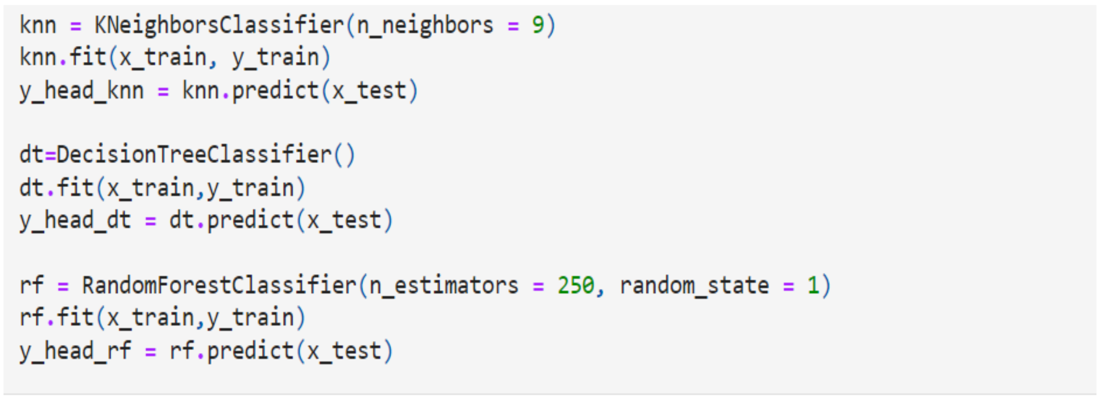
```
def my_confusion_matrix(y_test, y_pred, plt_title, accuracy_title):
    cm=confusion_matrix(y_test, y_pred)
    print(f'{accuracy_title} accuracy score:', '{:.2%}'.format(accuracy_score(y_test, y_pred)))
    print(classification_report(y_test, y_pred))
    sns.heatmap(cm, annot=True, fmt='g', cbar=False, cmap='BuPu')
    plt.xlabel('Predicted Values')
    plt.ylabel('Actual Values')
    plt.title(plt_title)
    plt.show()
    return cm
```

```
random_state = 42
classifier = [KNeighborsClassifier(),
              DecisionTreeClassifier(random_state = random_state),
             RandomForestClassifier(random_state = random_state),
             ]

knn_param_grid = {"n_neighbors": np.linspace(1,19,10, dtype = int).tolist(),
                 "weights": ["uniform","distance"],
                 "metric":["euclidean","manhattan"]}

dt_param_grid = {"min_samples_split" : range(10,500,20),
                "max_depth": range(1,20,2)}


rf_param_grid = {"max_features": [1,3,10],
                "min_samples_split":[2,3,10],
                "min_samples_leaf":[1,3,10],
               "bootstrap":[False],
                "n_estimators":[100,300],
                "criterion":["gini"]}
classifier_param = [knn_param_grid,
                     dt_param_grid,
                    rf_param_grid,
                     ]
```

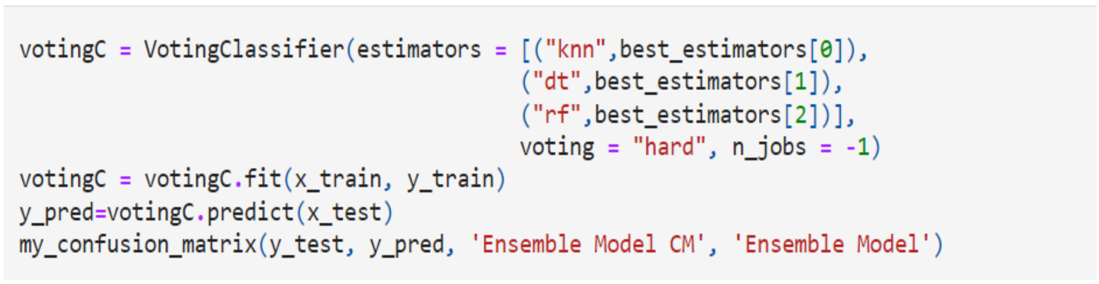
```
cv_result = []
best_estimators = []
for i in range(len(classifier)):
    clf = GridSearchCV(classifier[i], param_grid=classifier_param[i], cv = StratifiedKFold(n_splits = 10), scoring = "accuracy", n_jobs = -1,verbose
    clf.fit(x_train,y_train)
    cv_result.append(clf.best_score_ * 100)
    best_estimators.append(clf.best_estimator_)
    print(cv_result[i])
```



```python
cv_results = pd.DataFrame({"Cross Validation Means":cv_result, "ML Models":[ "KNeighborsClassifier", "Decision Tree Classifier",
             "Random Forest Classifier",
            ]})

g = sns.barplot("Cross Validation Means", "ML Models", data = cv_results)
g.set_xlabel("Mean Accuracy")
g.set_title("Cross Validation Scores")
```



```python
knn = KNeighborsClassifier(n_neighbors = 9)
knn.fit(x_train, y_train)
y_head_knn = knn.predict(x_test)

dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_head_dt = dt.predict(x_test)

rf = RandomForestClassifier(n_estimators = 250, random_state = 1)
rf.fit(x_train,y_train)
y_head_rf = rf.predict(x_test)
```



```python
votingC = VotingClassifier(estimators = [("knn",best_estimators[0]),
                                        ("dt",best_estimators[1]),
                                        ("rf",best_estimators[2])],
                                        voting = "hard", n_jobs = -1)
votingC = votingC.fit(x_train, y_train)
y_pred=votingC.predict(x_test)
my_confusion_matrix(y_test, y_pred, 'Ensemble Model CM', 'Ensemble Model')
```