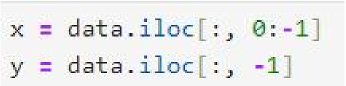
Model Development Phase

Date	11 July 2024
Team ID	SWTID1720013031
Project Title	Prediction and Analysis of Liver Patient Data Using Machine Learning
Maximum Marks	4 Marks

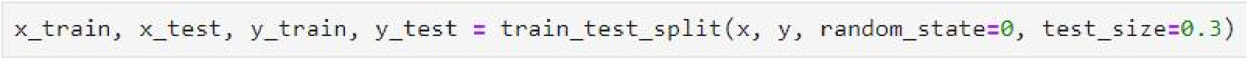
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

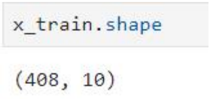
Initial Model Training Code:



```
x = data.iloc[:, 0:-1]
y = data.iloc[:, -1]
```

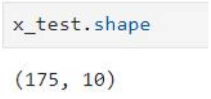


```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, test_size=0.3)
```



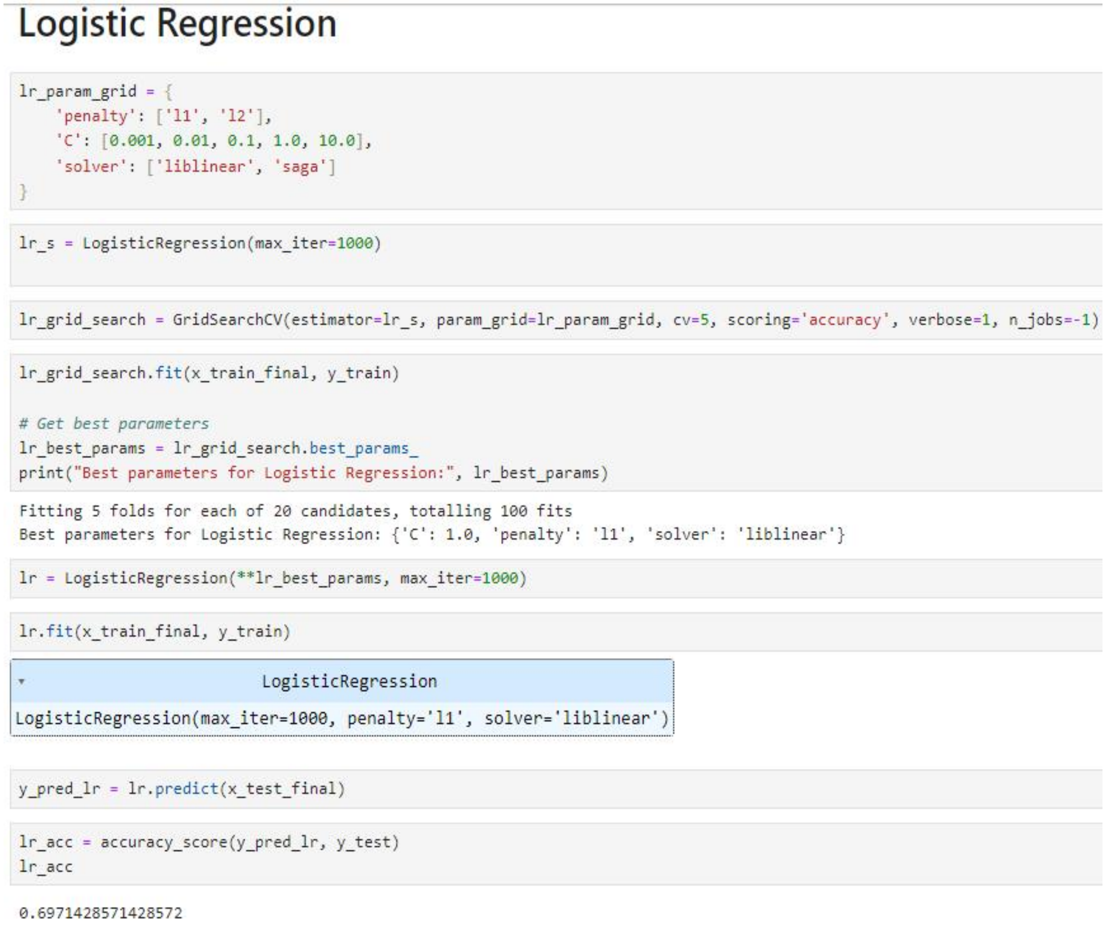
```
x_train.shape
```

```
(408, 10)
```



```
x_test.shape
```

```
(175, 10)
```



Logistic Regression

```
lr_param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1.0, 10.0],
    'solver': ['liblinear', 'saga']
}
```

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

```
lr_grid_search = GridSearchCV(estimator=lr_s, param_grid=lr_param_grid, cv=5, scoring='accuracy', verbose=1, n_jobs=-1)
```

```
lr_grid_search.fit(x_train_final, y_train)

# Get best parameters
lr_best_params = lr_grid_search.best_params_
print("Best parameters for Logistic Regression:", lr_best_params)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best parameters for Logistic Regression: {'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}
```

```
lr = LogisticRegression(**lr_best_params, max_iter=1000)
```

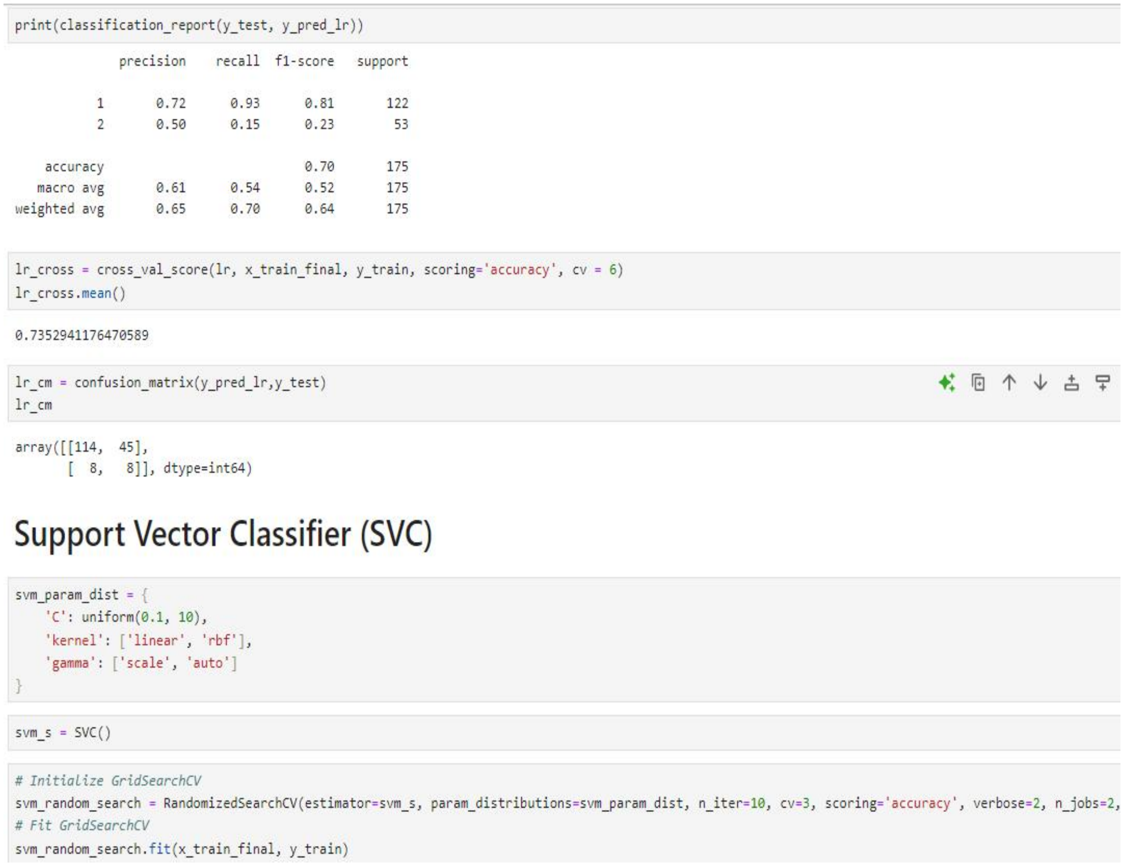
```
lr.fit(x_train_final, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, penalty='l1', solver='liblinear')
```

```
y_pred_lr = lr.predict(x_test_final)
```

```
lr_acc = accuracy_score(y_pred_lr, y_test)
lr_acc
```

```
0.6971428571428572
```



```
print(classification_report(y_test, y_pred_lr))
```

```
              precision    recall  f1-score   support

           1       0.72      0.93      0.81       122
           2       0.50      0.15      0.23        53

    accuracy                           0.70       175
   macro avg       0.61      0.54      0.52       175
weighted avg       0.65      0.70      0.64       175
```

```
lr_cross = cross_val_score(lr, x_train_final, y_train, scoring='accuracy', cv = 6)
lr_cross.mean()
```

```
0.7352941176470589
```

```
lr_cm = confusion_matrix(y_pred_lr,y_test)
lr_cm
```

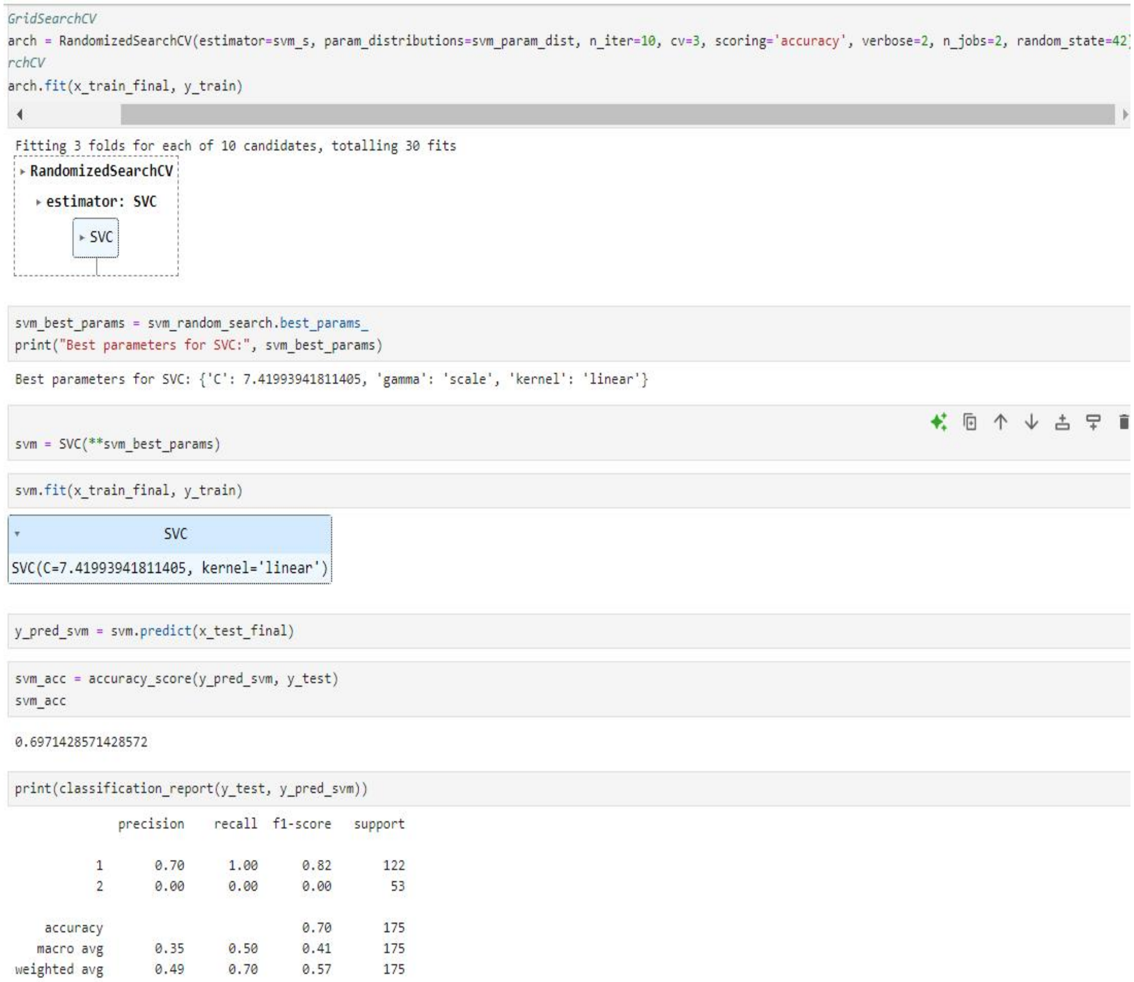
```
array([[114,  45],
       [  8,   8]], dtype=int64)
```

Support Vector Classifier (SVC)

```
svm_param_dist = {
    'C': uniform(0.1, 10),
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}
```

```
svm_s = SVC()
```

```
# Initialize GridSearchCV
svm_random_search = RandomizedSearchCV(estimator=svm_s, param_distributions=svm_param_dist, n_iter=10, cv=3, scoring='accuracy', verbose=2, n_jobs=2,
# Fit GridSearchCV
svm_random_search.fit(x_train_final, y_train)
```



```
GridSearchCV
arch = RandomizedSearchCV(estimator=svm_s, param_distributions=svm_param_dist, n_iter=10, cv=3, scoring='accuracy', verbose=2, n_jobs=2, random_state=42)
rchCV
arch.fit(x_train_final, y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
svm_best_params = svm_random_search.best_params_
print("Best parameters for SVC:", svm_best_params)
```

```
Best parameters for SVC: {'C': 7.41993941811405, 'gamma': 'scale', 'kernel': 'linear'}
```

```
svm = SVC(**svm_best_params)
```

```
svm.fit(x_train_final, y_train)
```

```
SVC(C=7.41993941811405, kernel='linear')
```

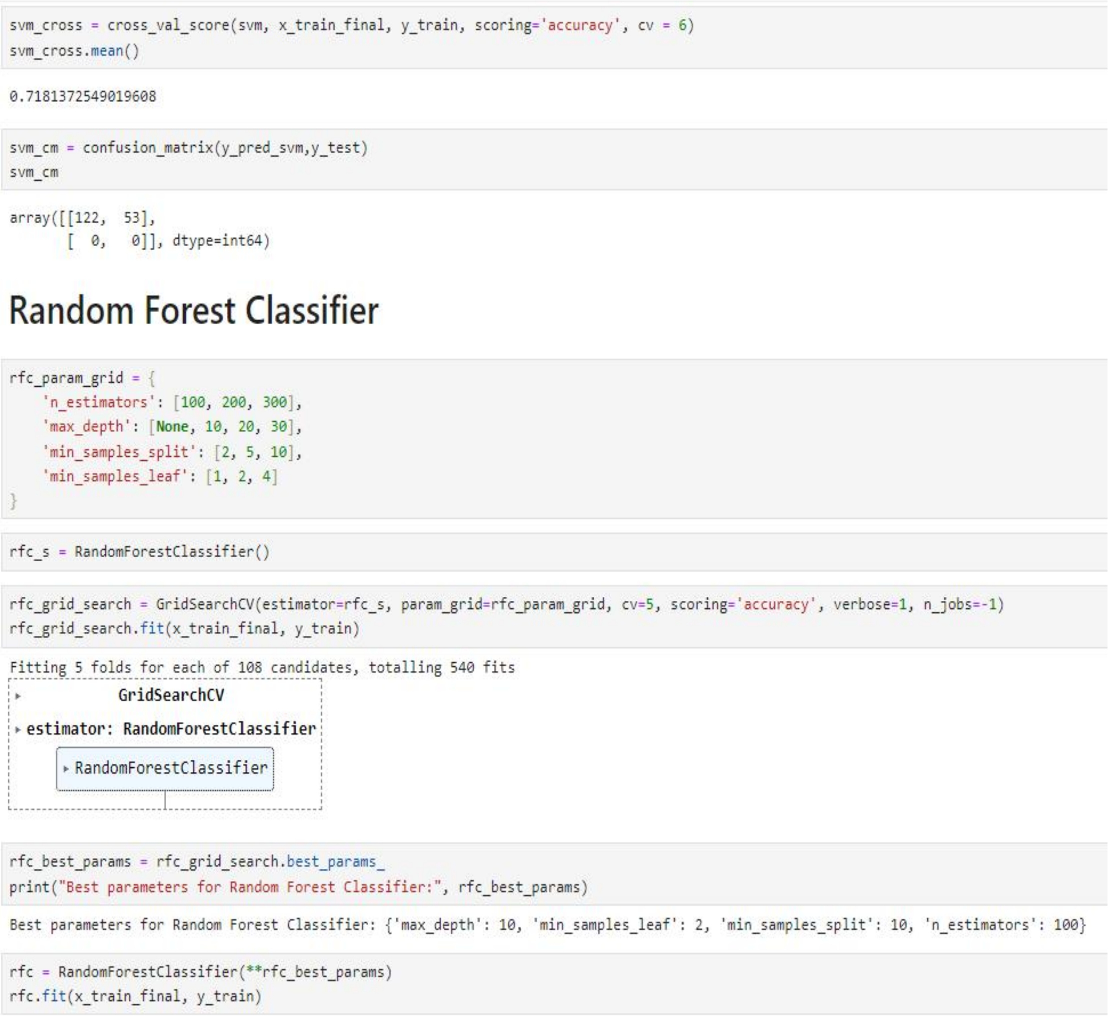
```
y_pred_svm = svm.predict(x_test_final)
```

```
svm_acc = accuracy_score(y_pred_svm, y_test)
svm_acc
```

```
0.6971428571428572
```

```
print(classification_report(y_test, y_pred_svm))
```

```
              precision    recall  f1-score   support

           1       0.70      1.00      0.82       122
           2       0.00      0.00      0.00        53

    accuracy                           0.70       175
   macro avg       0.35      0.50      0.41       175
weighted avg       0.49      0.70      0.57       175
```



```
svm_cross = cross_val_score(svm, x_train_final, y_train, scoring='accuracy', cv = 6)
svm_cross.mean()
```

```
0.7181372549019608
```

```
svm_cm = confusion_matrix(y_pred_svm,y_test)
svm_cm
```

```
array([[122,  53],
       [  0,   0]], dtype=int64)
```

Random Forest Classifier

```
rfc_param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
rfc_s = RandomForestClassifier()
```

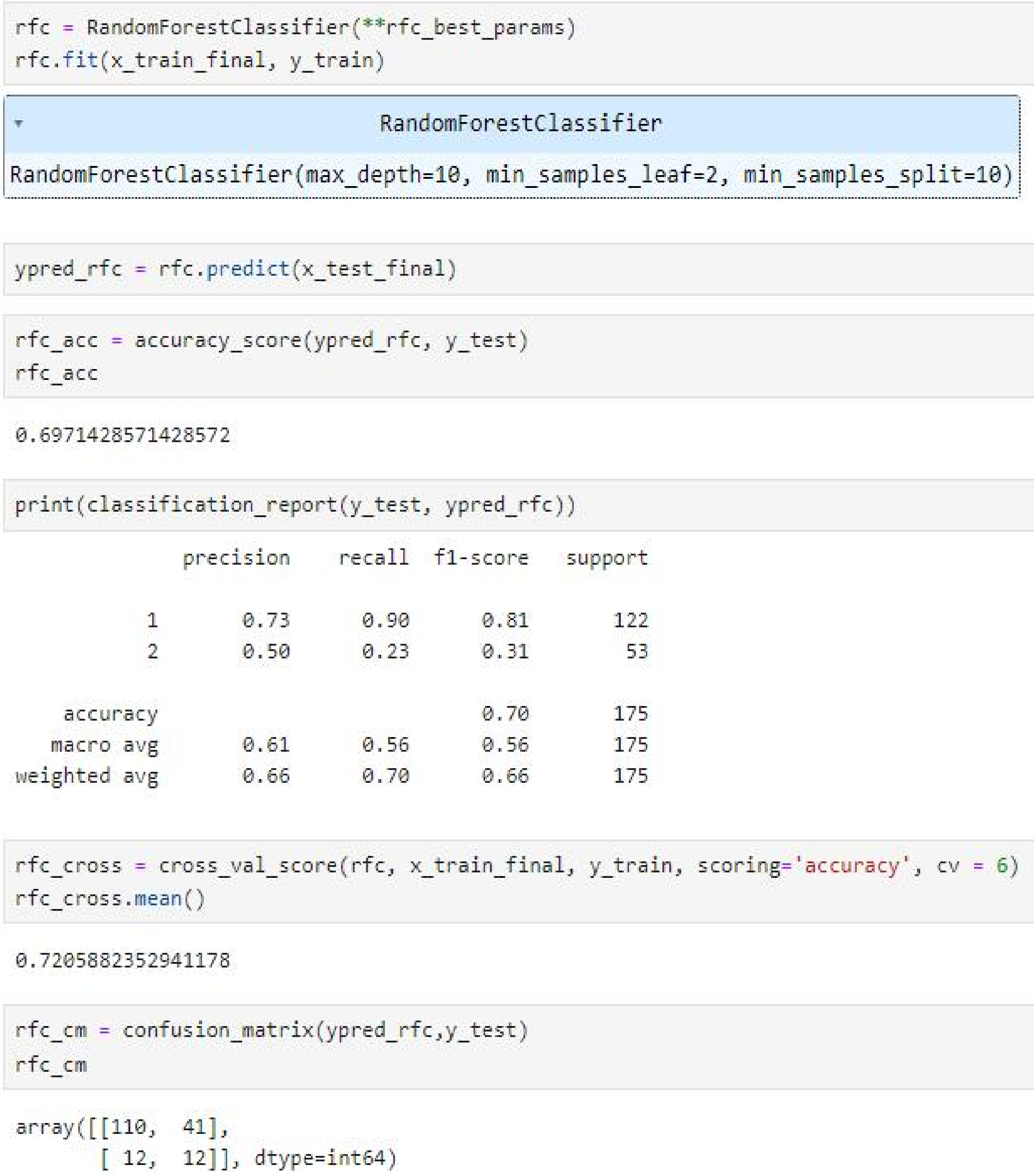
```
rfc_grid_search = GridSearchCV(estimator=rfc_s, param_grid=rfc_param_grid, cv=5, scoring='accuracy', verbose=1, n_jobs=-1)
rfc_grid_search.fit(x_train_final, y_train)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
```

```
rfc_best_params = rfc_grid_search.best_params_
print("Best parameters for Random Forest Classifier:", rfc_best_params)
```

```
Best parameters for Random Forest Classifier: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 100}
```

```
rfc = RandomForestClassifier(**rfc_best_params)
rfc.fit(x_train_final, y_train)
```



```
rfc = RandomForestClassifier(**rfc_best_params)
rfc.fit(x_train_final, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=10, min_samples_leaf=2, min_samples_split=10)
```

```
ypred_rfc = rfc.predict(x_test_final)
```

```
rfc_acc = accuracy_score(ypred_rfc, y_test)
rfc_acc
```

```
0.6971428571428572
```

```
print(classification_report(y_test, ypred_rfc))
```

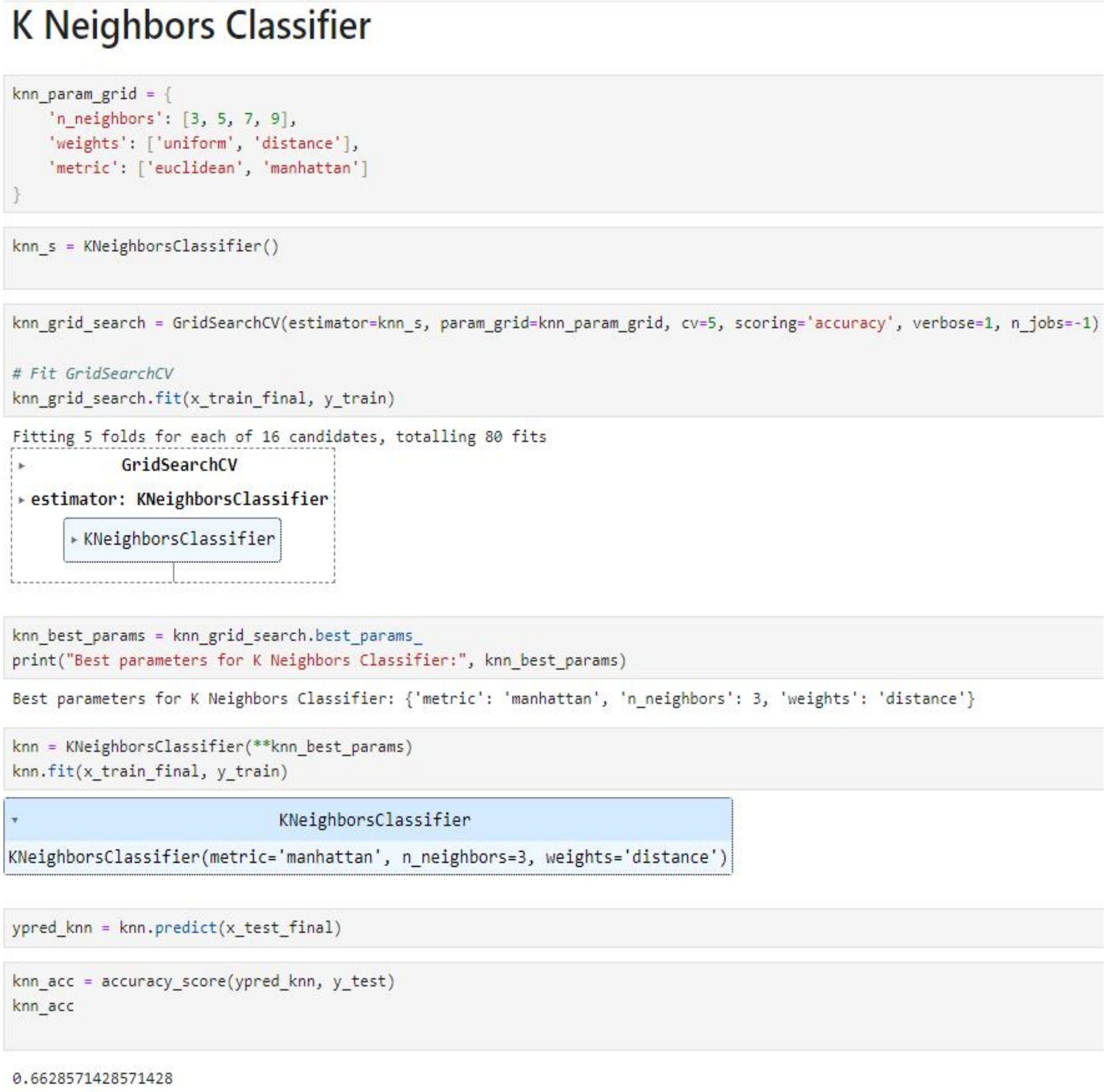
```
              precision    recall  f1-score   support

           1       0.73      0.90      0.81       122
           2       0.50      0.23      0.31        53

    accuracy                           0.70       175
   macro avg       0.61      0.56      0.56       175
weighted avg       0.66      0.70      0.66       175
```

```
rfc_cross = cross_val_score(rfc, x_train_final, y_train, scoring='accuracy', cv = 6)
rfc_cross.mean()
```

```
0.7205882352941178
```

```
rfc_cm = confusion_matrix(ypred_rfc,y_test)
rfc_cm
```

```
array([[110,  41],
       [ 12,  12]], dtype=int64)
```



K Neighbors Classifier

```
knn_param_grid = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
```

```
knn_s = KNeighborsClassifier()
```

```
knn_grid_search = GridSearchCV(estimator=knn_s, param_grid=knn_param_grid, cv=5, scoring='accuracy', verbose=1, n_jobs=-1)

# Fit GridSearchCV
knn_grid_search.fit(x_train_final, y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
```

```
knn_best_params = knn_grid_search.best_params_
print("Best parameters for K Neighbors Classifier:", knn_best_params)
```

```
Best parameters for K Neighbors Classifier: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
```

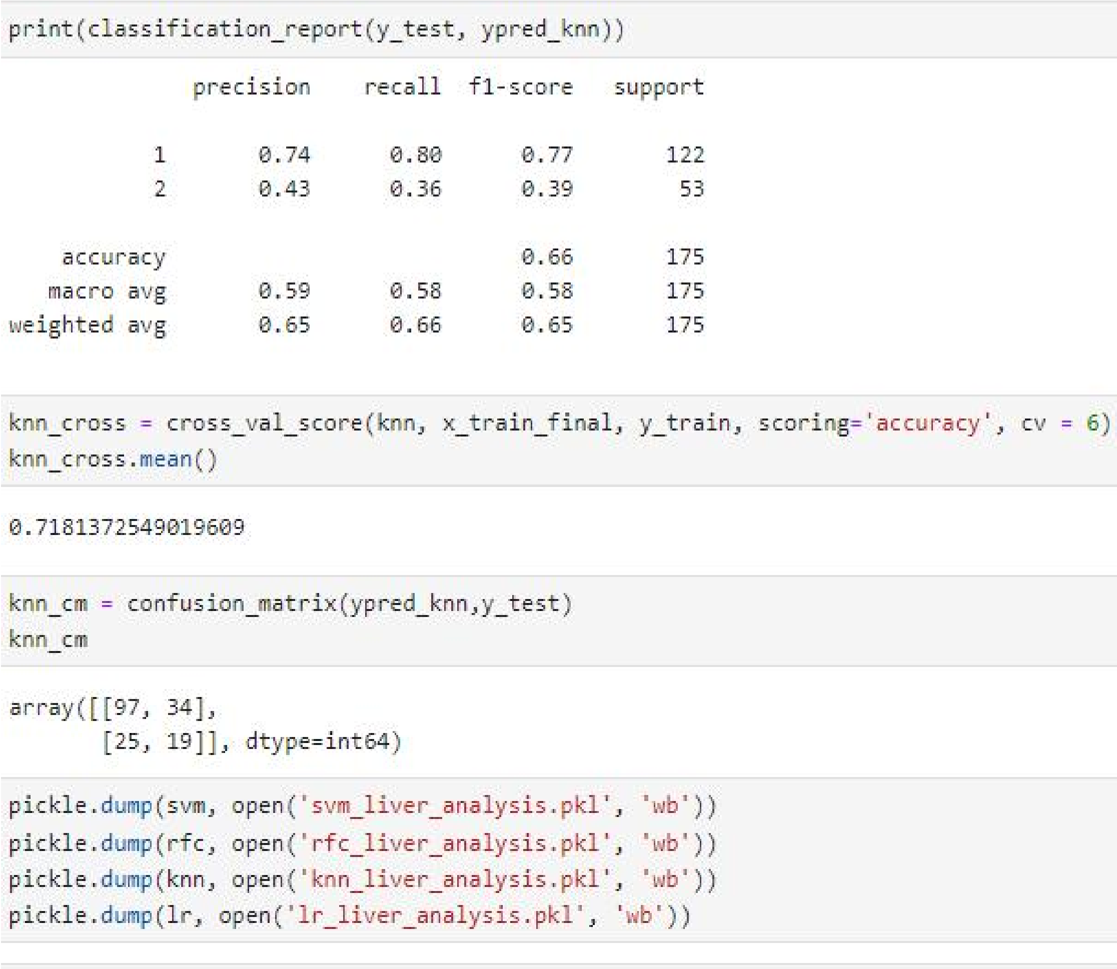
```
knn = KNeighborsClassifier(**knn_best_params)
knn.fit(x_train_final, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(metric='manhattan', n_neighbors=3, weights='distance')
```

```
ypred_knn = knn.predict(x_test_final)
```

```
knn_acc = accuracy_score(ypred_knn, y_test)
knn_acc
```

```
0.6628571428571428
```



```
print(classification_report(y_test, ypred_knn))
```

```
              precision    recall  f1-score   support

           1       0.74      0.80      0.77       122
           2       0.43      0.36      0.39        53

    accuracy                           0.66       175
   macro avg       0.59      0.58      0.58       175
weighted avg       0.65      0.66      0.65       175
```

```
knn_cross = cross_val_score(knn, x_train_final, y_train, scoring='accuracy', cv = 6)
knn_cross.mean()
```

```
0.7181372549019609
```

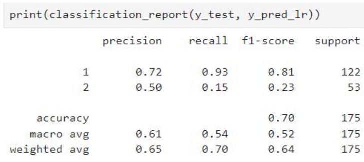
```
knn_cm = confusion_matrix(ypred_knn,y_test)
knn_cm
```

```
array([[97, 34],
       [25, 19]], dtype=int64)
```

```
pickle.dump(svm, open('svm_liver_analysis.pkl', 'wb'))
pickle.dump(rfc, open('rfc_liver_analysis.pkl', 'wb'))
pickle.dump(knn, open('knn_liver_analysis.pkl', 'wb'))
pickle.dump(lr, open('lr_liver_analysis.pkl', 'wb'))
```

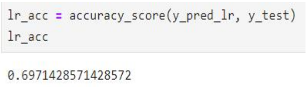
Model Validation and Evaluation Report:

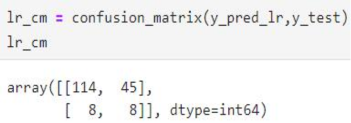
Model	Classification Report	Accuracy	Confusion Matrix

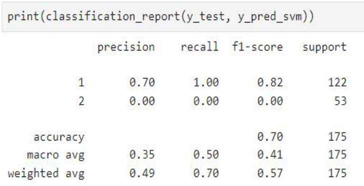
<table>
<tr>
<td>Logistic Regression</td>
<td>
<pre>
print(classification_report(y_test, y_pred_lr))

              precision   recall f1-score   support

           1   0.72   0.93   0.81   122
           2   0.50   0.15   0.23   53

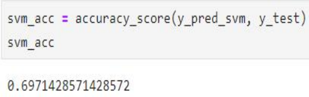
    accuracy   0.70   175
   macro avg   0.61   0.54   0.52   175
weighted avg   0.65   0.70   0.64   175
</pre></td>
<td>69.71%

<pre>
lr_acc = accuracy_score(y_pred_lr, y_test)
lr_acc

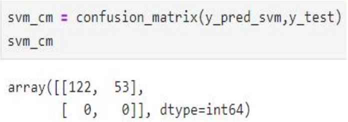
0.6971428571428572
</pre></td>
<td>
<pre>
lr_cm = confusion_matrix(y_pred_lr,y_test)
lr_cm

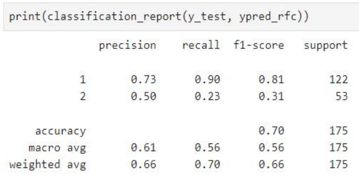
array([[114, 45],
       [8,   8]], dtype=int64)
</pre></td>
</tr>
<tr>
<td>Support Vector Classifier (SVC)</td>
<td>
<pre>
print(classification_report(y_test, y_pred_svm))

              precision   recall f1-score   support

           1   0.70   1.00   0.82   122
           2   0.00   0.00   0.00   53

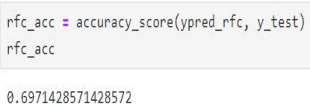
    accuracy   0.70   175
   macro avg   0.35   0.50   0.41   175
weighted avg   0.49   0.70   0.57   175
</pre></td>
<td>69.71%

<pre>
svm_acc = accuracy_score(y_pred_svm, y_test)
svm_acc

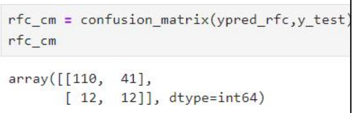
0.6971428571428572
</pre></td>
<td>
<pre>
svm_cm = confusion_matrix(y_pred_svm,y_test)
svm_cm

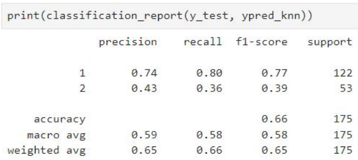
array([[122, 53],
       [0,   0]], dtype=int64)
</pre></td>
</tr>
<tr>
<td>Random Forest Classifier</td>
<td>
<pre>
print(classification_report(y_test, ypred_rfc))

              precision   recall f1-score   support

           1   0.73   0.90   0.81   122
           2   0.50   0.23   0.31   53

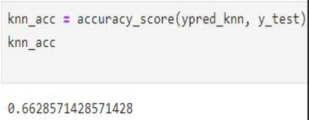
    accuracy   0.70   175
   macro avg   0.61   0.56   0.56   175
weighted avg   0.66   0.70   0.66   175
</pre></td>
<td>69.7%

<pre>
rfc_acc = accuracy_score(ypred_rfc, y_test)
rfc_acc

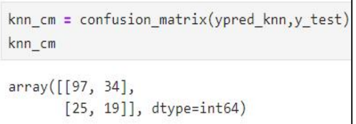
0.6971428571428572
</pre></td>
<td>
<pre>
rfc_cm = confusion_matrix(ypred_rfc,y_test)
rfc_cm

array([[110, 41],
       [12, 12]], dtype=int64)
</pre></td>
</tr>
<tr>
<td>K Neighbors Classifier</td>
<td>
<pre>
print(classification_report(y_test, ypred_knn))

              precision   recall f1-score   support

           1   0.74   0.80   0.77   122
           2   0.43   0.36   0.39   53

    accuracy   0.66   175
   macro avg   0.59   0.58   0.58   175
weighted avg   0.65   0.66   0.65   175
</pre></td>
<td>66.2%

<pre>
knn_acc = accuracy_score(ypred_knn, y_test)
knn_acc

0.6628571428571428
</pre></td>
<td>
<pre>
knn_cm = confusion_matrix(ypred_knn,y_test)
knn_cm

array([[97, 34],
       [25, 19]], dtype=int64)
</pre></td>
</tr>
</table>