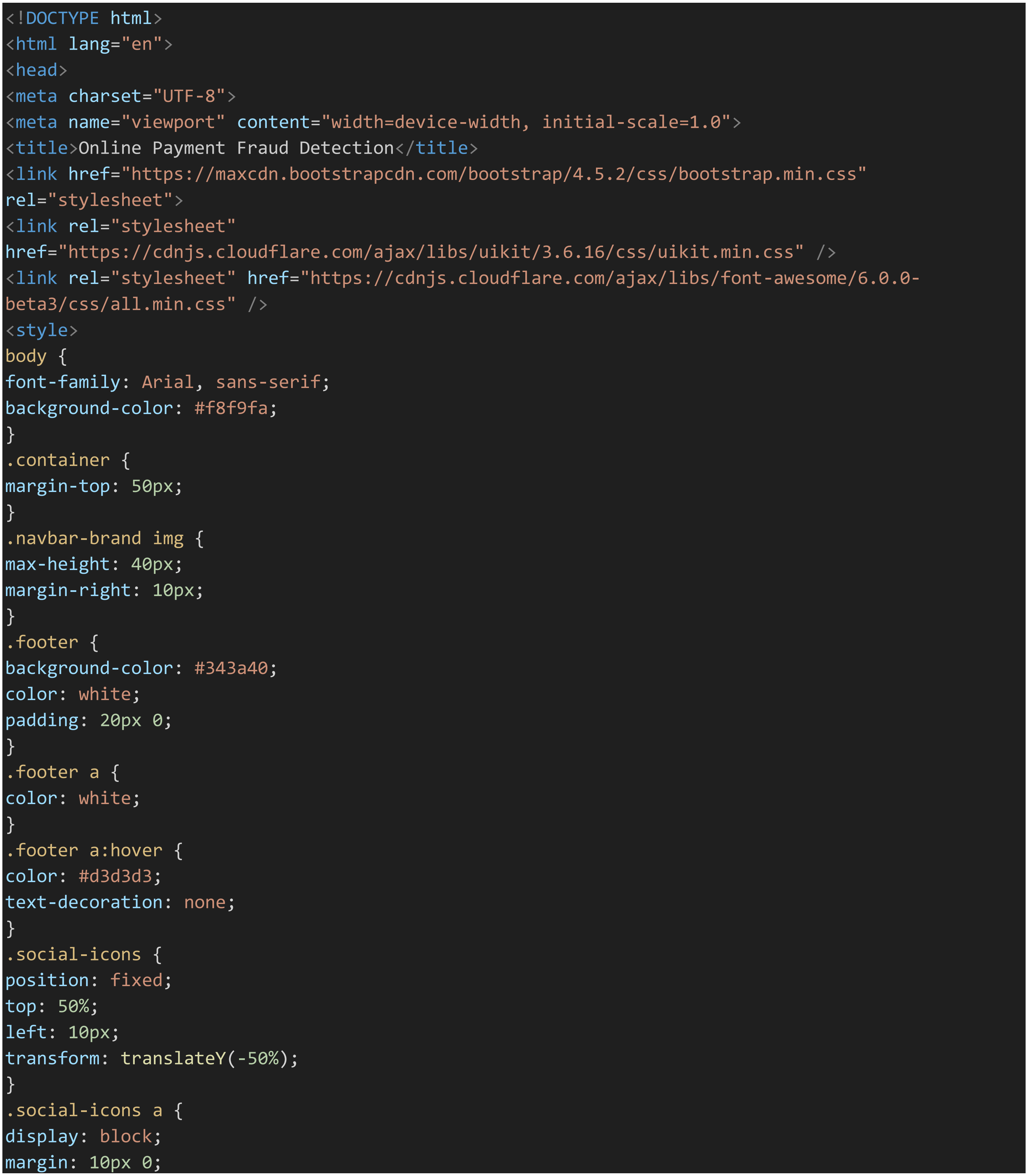
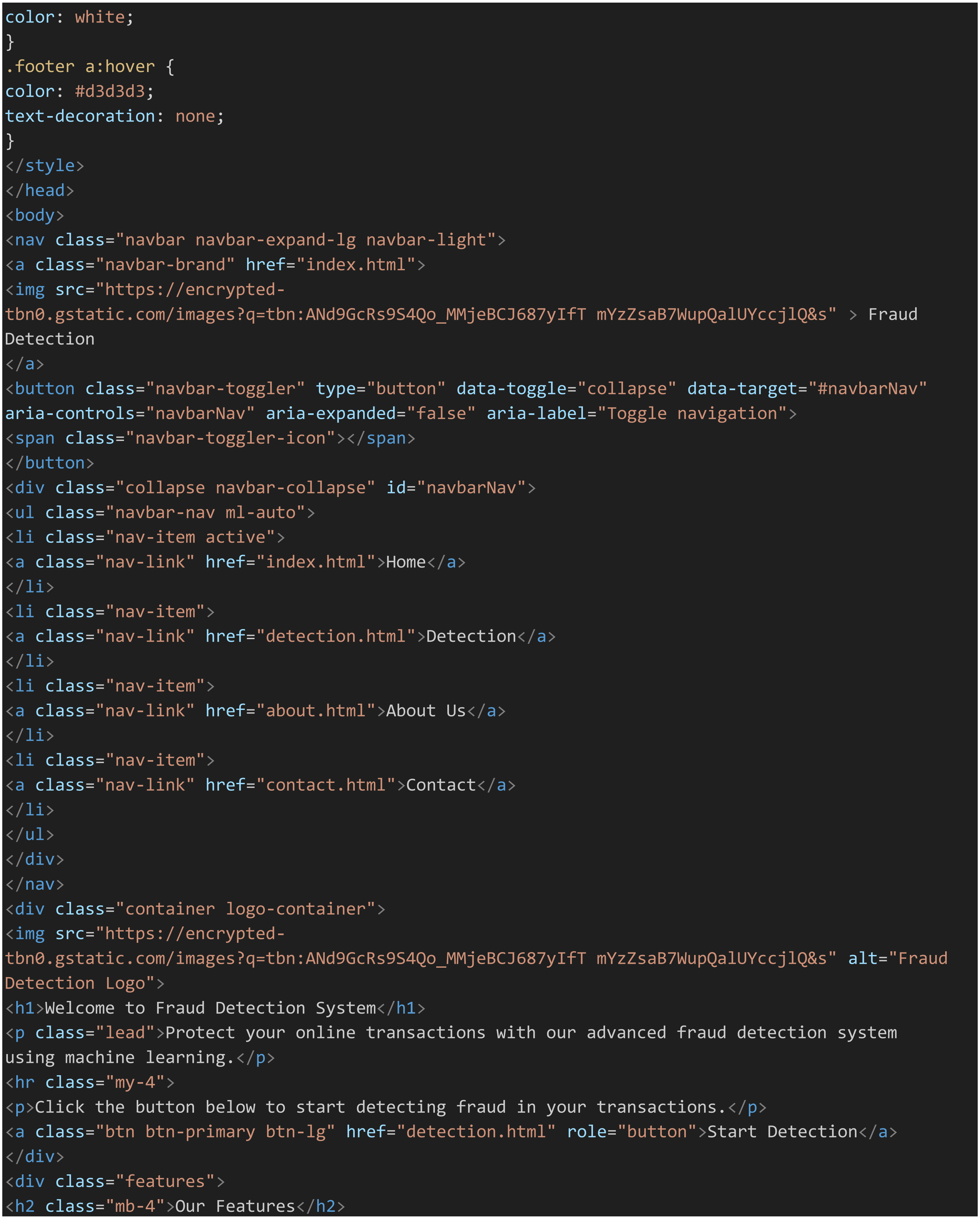
Model Traning File and Testing File

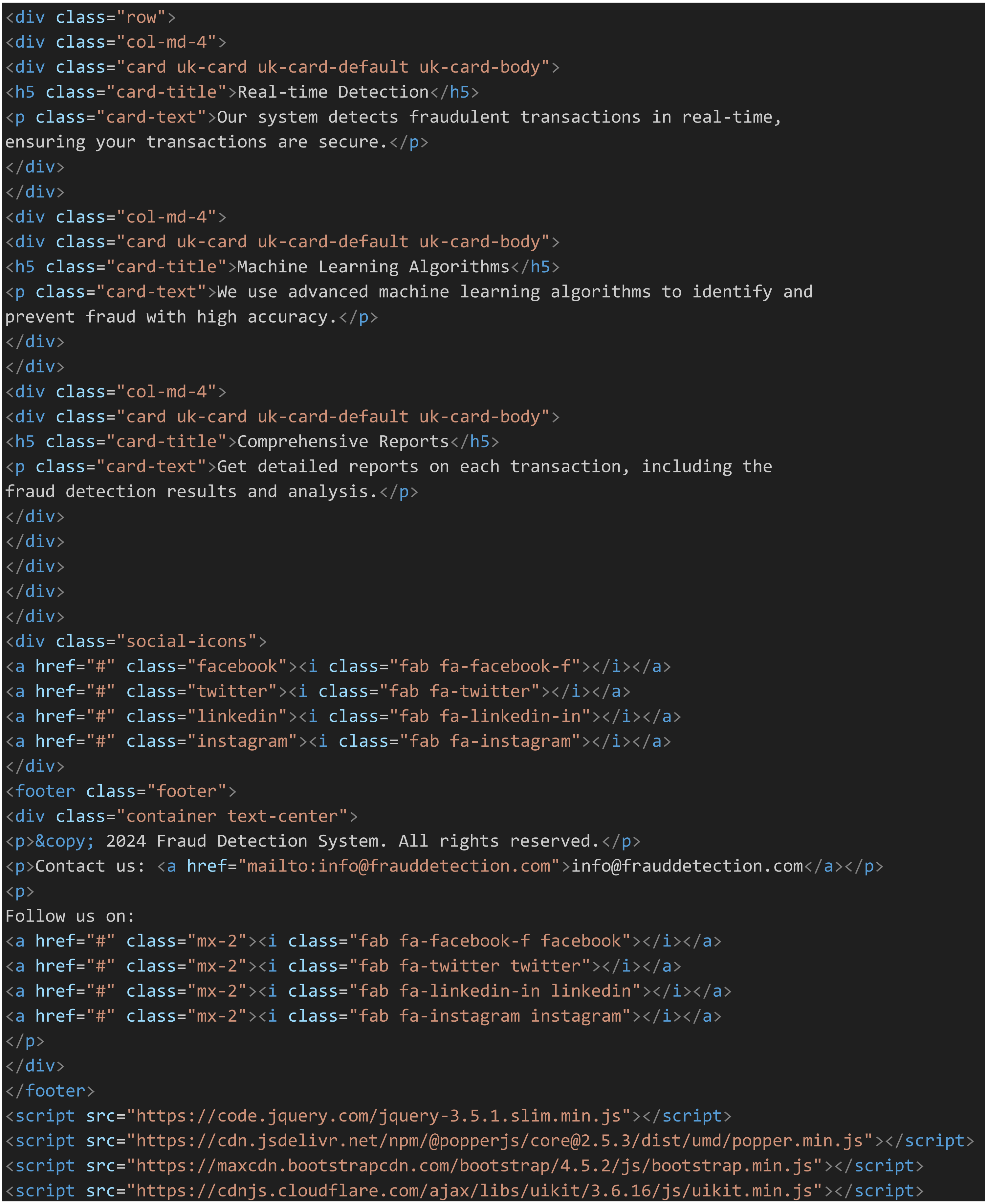


```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Online Payment Fraud Detection</title>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.6.16/css/uikit.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css" />
<style>
body {
font-family: Arial, sans-serif;
background-color: #f8f9fa;
}
.container {
margin-top: 50px;
}
.navbar-brand img {
max-height: 40px;
margin-right: 10px;
}
.footer {
background-color: #343a40;
color: white;
padding: 20px 0;
}
.footer a {
color: white;
}
.footer a:hover {
color: #d3d3d3;
text-decoration: none;
}
.social-icons {
position: fixed;
top: 50%;
left: 10px;
transform: translateY(-50%);
}
.social-icons a {
display: block;
margin: 10px 0;
```

```css
font-size: 24px;
}
.social-icons a.facebook {
color: #3b5998;
}
.social-icons a.twitter {
color: #1da1f2;
}
.social-icons a.linkedin {
color: #0077b5;
}
.social-icons a.instagram {
color: #e4405f;
}
.features {
margin-top: 50px;
}
body {
font-family: Arial, sans-serif;
background-color: #f8f9fa;
}
.navbar {
background-color: #004085; /* Dark blue background */
}
.navbar-brand, .navbar-nav .nav-link {
color: #ffffff !important; /* White text color */
}
.navbar-brand img {
max-height: 40px;
margin-right: 10px;
}
.logo-container {
text-align: center;
margin-top: 20px;
}
.logo-container img {
max-width: 150px;
}
.logo-container h1 {
font-size: 2em;
color: #004085;
margin-top: 10px;
}
.footer {
background-color: #343a40;
color: white;
padding: 20px 0;
}
.footer a {
```

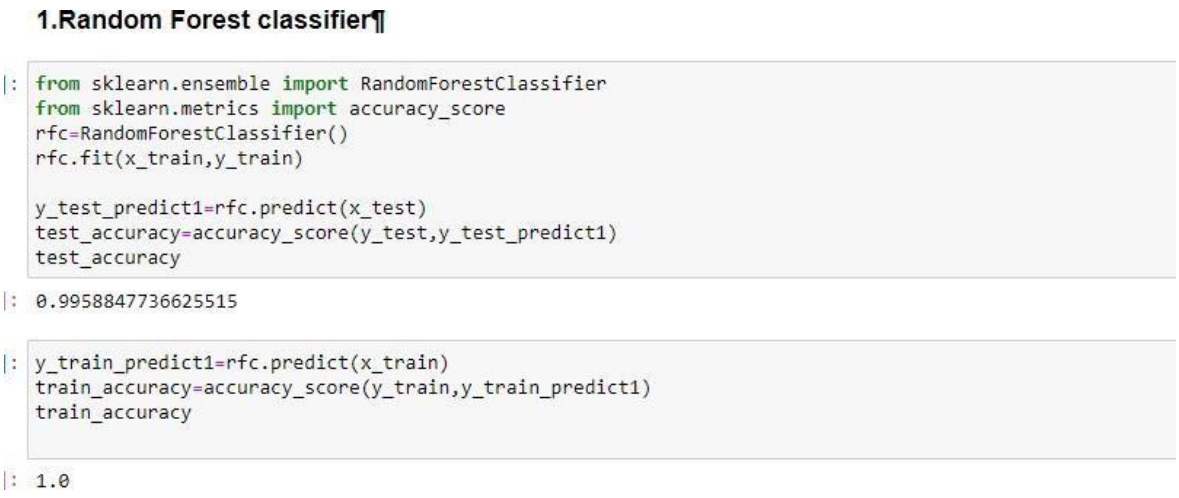


```
color: white;
}
.footer a:hover {
color: #d3d3d3;
text-decoration: none;
}
</style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light">
<a class="navbar-brand" href="index.html">
<img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRs9S4Qo_MMjeBCJ687yIfT mYzZsaB7WupQalUYccjlQ&s" > Fraud
Detection
</a>
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav"
aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav ml-auto">
<li class="nav-item active">
<a class="nav-link" href="index.html">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="detection.html">Detection</a>
</li>
<li class="nav-item">
<a class="nav-link" href="about.html">About Us</a>
</li>
<li class="nav-item">
<a class="nav-link" href="contact.html">Contact</a>
</li>
</ul>
</div>
</nav>
<div class="container logo-container">
<img src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcRs9S4Qo_MMjeBCJ687yIfT mYzZsaB7WupQalUYccjlQ&s" alt="Fraud
Detection Logo">
<h1>Welcome to Fraud Detection System</h1>
<p class="lead">Protect your online transactions with our advanced fraud detection system
using machine learning.</p>
<hr class="my-4">
<p>Click the button below to start detecting fraud in your transactions.</p>
<a class="btn btn-primary btn-lg" href="detection.html" role="button">Start Detection</a>
</div>
<div class="features">
<h2 class="mb-4">Our Features</h2>
```



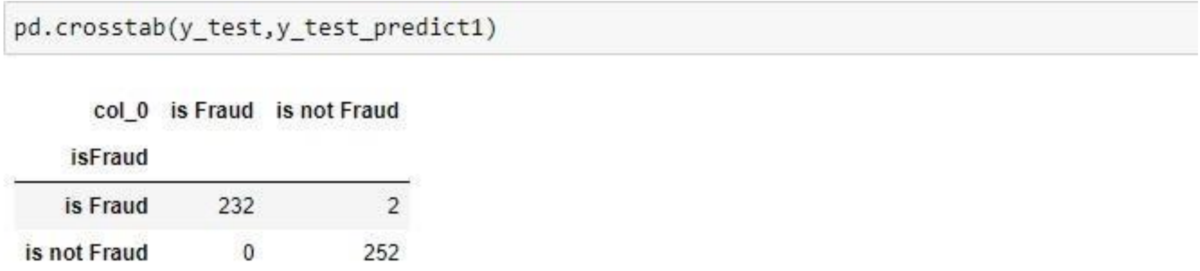
```
<div class="row">
<div class="col-md-4">
<div class="card uk-card uk-card-default uk-card-body">
<h5 class="card-title">Real-time Detection</h5>
<p class="card-text">Our system detects fraudulent transactions in real-time,
ensuring your transactions are secure.</p>
</div>
</div>
<div class="col-md-4">
<div class="card uk-card uk-card-default uk-card-body">
<h5 class="card-title">Machine Learning Algorithms</h5>
<p class="card-text">We use advanced machine learning algorithms to identify and
prevent fraud with high accuracy.</p>
</div>
</div>
<div class="col-md-4">
<div class="card uk-card uk-card-default uk-card-body">
<h5 class="card-title">Comprehensive Reports</h5>
<p class="card-text">Get detailed reports on each transaction, including the
fraud detection results and analysis.</p>
</div>
</div>
</div>
</div>
</div>
<div class="social-icons">
<a href="#" class="facebook"><i class="fab fa-facebook-f"></i></a>
<a href="#" class="twitter"><i class="fab fa-twitter"></i></a>
<a href="#" class="linkedin"><i class="fab fa-linkedin-in"></i></a>
<a href="#" class="instagram"><i class="fab fa-instagram"></i></a>
</div>
<footer class="footer">
<div class="container text-center">
<p>&copy; 2024 Fraud Detection System. All rights reserved.</p>
<p>Contact us: <a href="mailto:info@frauddetection.com">info@frauddetection.com</a></p>
<p>
Follow us on:
<a href="#" class="mx-2"><i class="fab fa-facebook-f facebook"></i></a>
<a href="#" class="mx-2"><i class="fab fa-twitter twitter"></i></a>
<a href="#" class="mx-2"><i class="fab fa-linkedin-in linkedin"></i></a>
<a href="#" class="mx-2"><i class="fab fa-instagram instagram"></i></a>
</p>
</div>
</footer>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.6.16/js/uikit.min.js"></script>
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.6.16/js/uikit icons.min.js"></script>
</body>
</html>
```



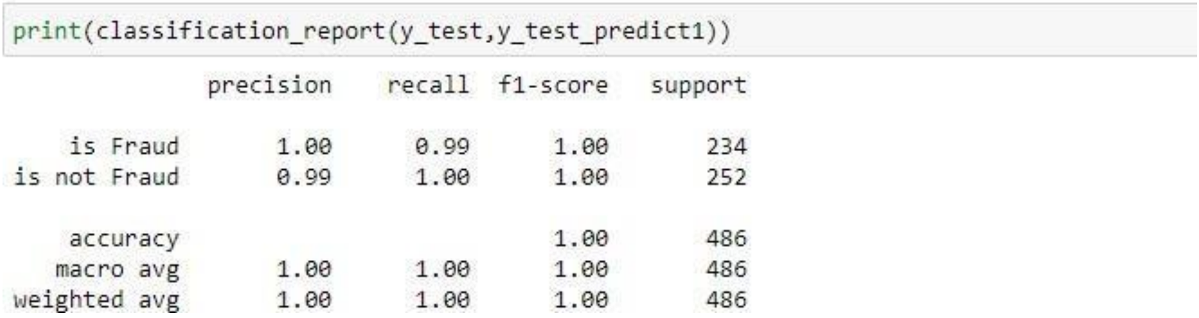
1.Random Forest classifier¶

```
[ ]: from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
     rfc=RandomForestClassifier()
     rfc.fit(x_train,y_train)

     y_test_predict1=rfc.predict(x_test)
     test_accuracy=accuracy_score(y_test,y_test_predict1)
     test_accuracy

[ ]: 0.9958847736625515

[ ]: y_train_predict1=rfc.predict(x_train)
     train_accuracy=accuracy_score(y_train,y_train_predict1)
     train_accuracy

[ ]: 1.0
```



```
pd.crosstab(y_test,y_test_predict1)
```

col_0 isFraud	is Fraud	is not Fraud
is Fraud	232	2
is not Fraud	0	252

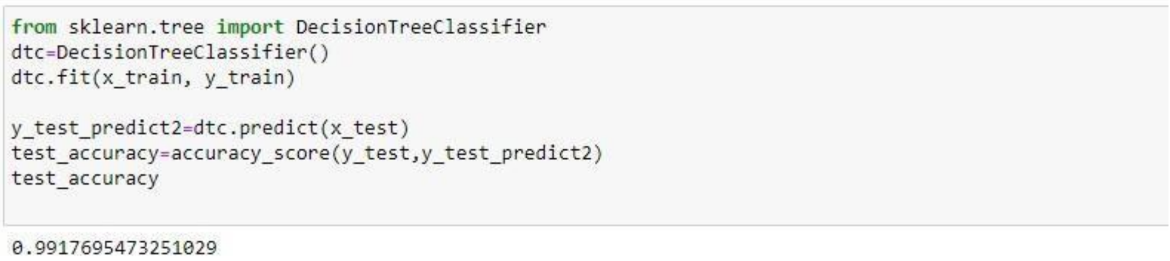


```
print(classification_report(y_test,y_test_predict1))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.99      1.00       234
is not Fraud       0.99      1.00      1.00       252

    accuracy                           1.00       486
   macro avg       1.00      1.00      1.00       486
weighted avg       1.00      1.00      1.00       486
```

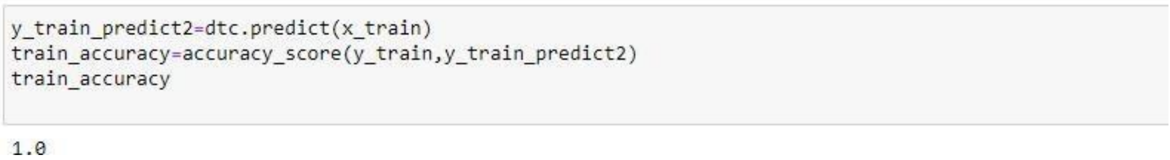
Decision Tree Classifier

A function named Decisiontree is created and train and test data are passed as the parameters. Inside the function, the DecisiontreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with the .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



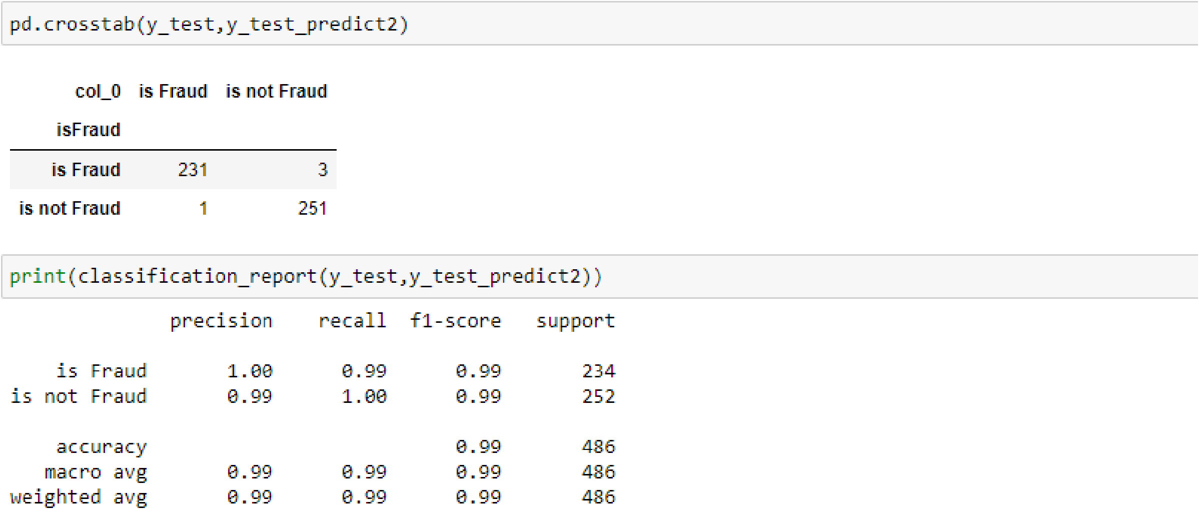
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train, y_train)

y_test_predict2=dtc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict2)
test_accuracy
```

```
0.9917695473251029
```



```
y_train_predict2=dtc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict2)
train_accuracy
```

```
1.0
```



```
pd.crosstab(y_test,y_test_predict2)
```

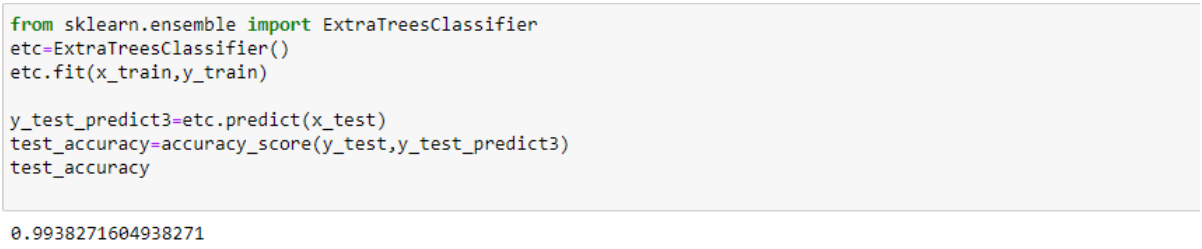
col_0	is Fraud	is not Fraud
isFraud		
is Fraud	231	3
is not Fraud	1	251

```
print(classification_report(y_test,y_test_predict2))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.99      0.99       234
is not Fraud       0.99      1.00      0.99       252

    accuracy                           0.99       486
   macro avg       0.99      0.99      0.99       486
weighted avg       0.99      0.99      0.99       486
```

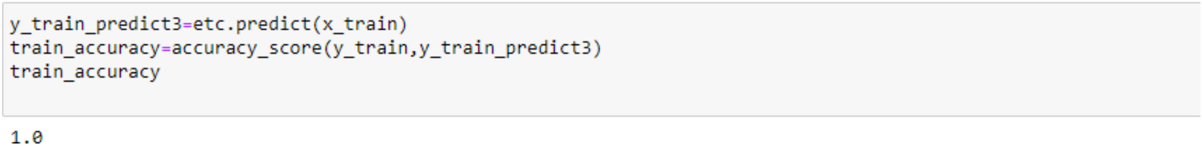
ExtraTrees Classifier

A function named ExtraTree is created and train and test data are passed as the parameters. Inside the function, ExtraTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



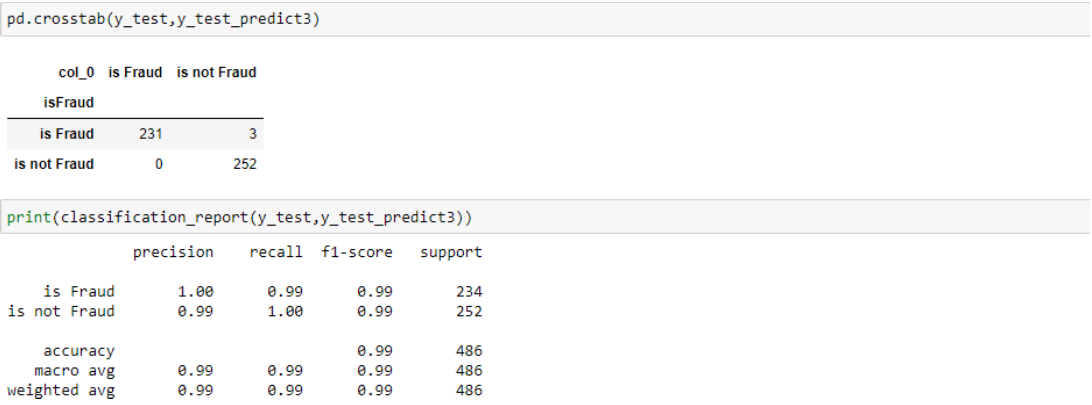
```
from sklearn.ensemble import ExtraTreesClassifier
etc=ExtraTreesClassifier()
etc.fit(x_train,y_train)

y_test_predict3=etc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict3)
test_accuracy
```

```
0.9938271604938271
```



```
y_train_predict3=etc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict3)
train_accuracy
```

```
1.0
```



```
pd.crosstab(y_test,y_test_predict3)
```

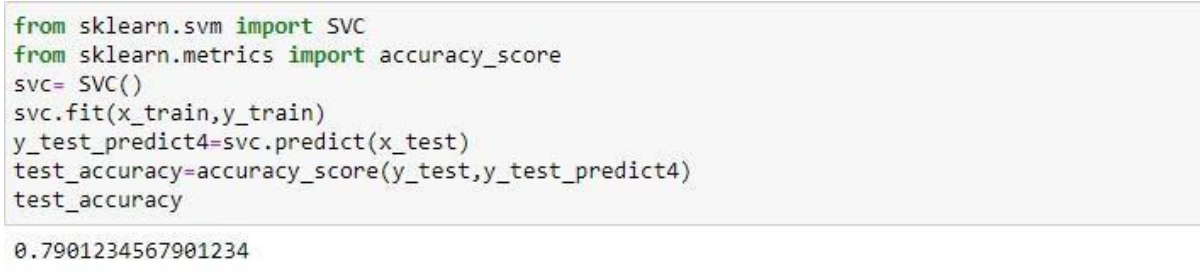
col_0	is Fraud	is not Fraud
isFraud		
is Fraud	231	3
is not Fraud	0	252

```
print(classification_report(y_test,y_test_predict3))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.99      0.99       234
is not Fraud       0.99      1.00      0.99       252

    accuracy                           0.99       486
   macro avg       0.99      0.99      0.99       486
weighted avg       0.99      0.99      0.99       486
```

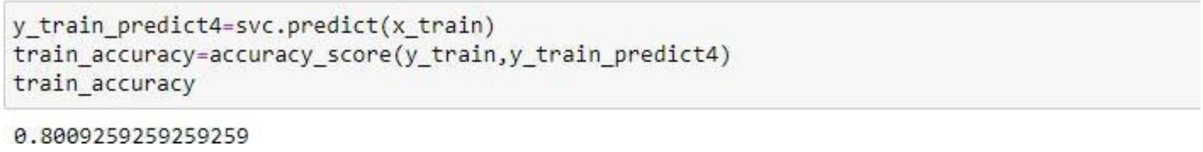
Support Vector Machine Classifier

A function named SupportVector is created and train and test data are passed as the parameters. Inside the function, the SupportVectorClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, confusion matrix and classification report is done



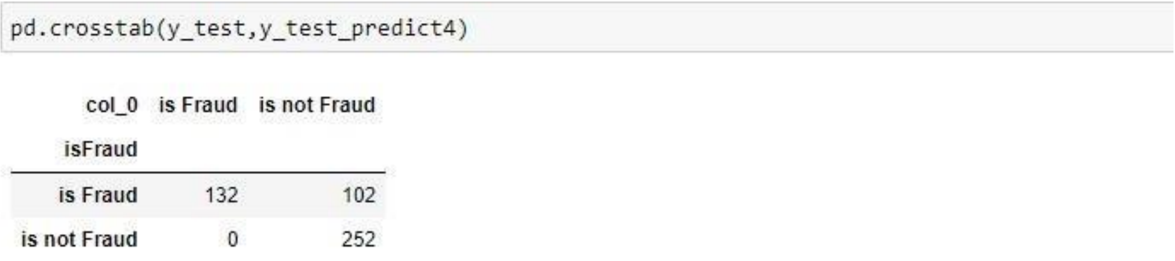
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC()
svc.fit(x_train,y_train)
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict4)
test_accuracy
```

```
0.7901234567901234
```



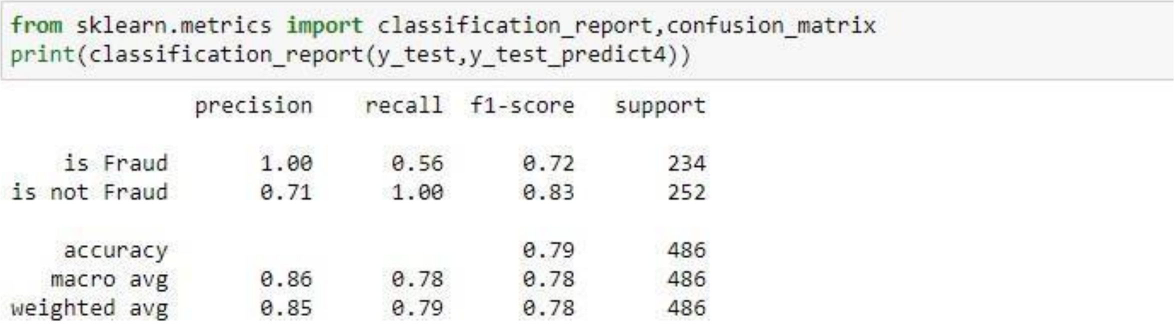
```
y_train_predict4=svc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict4)
train_accuracy
```

```
0.8009259259259259
```



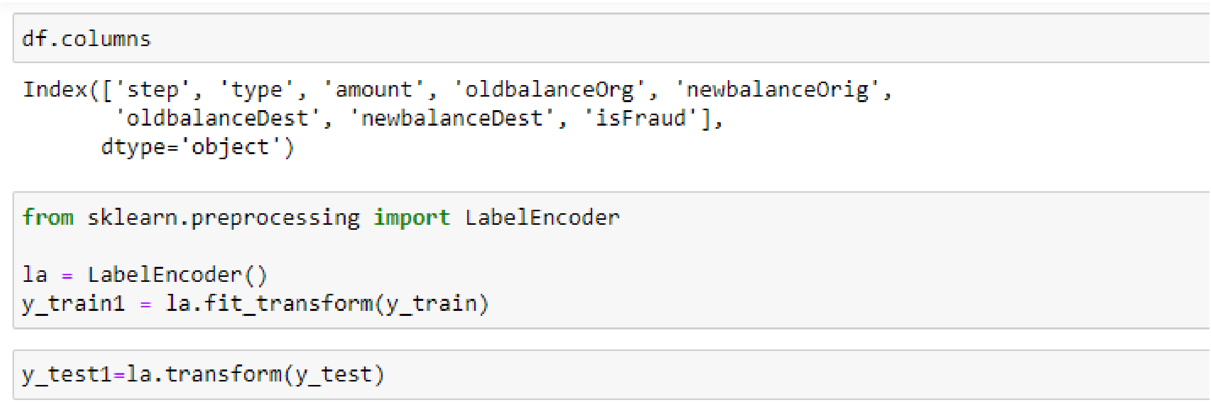
```
pd.crosstab(y_test,y_test_predict4)
```

col_0	is Fraud	is not Fraud
isFraud		
is Fraud	132	102
is not Fraud	0	252



```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_test_predict4))
```

```
              precision    recall  f1-score   support

    is Fraud       1.00      0.56      0.72       234
is not Fraud       0.71      1.00      0.83       252

    accuracy                           0.79       486
   macro avg       0.86      0.78      0.78       486
weighted avg       0.85      0.79      0.78       486
```



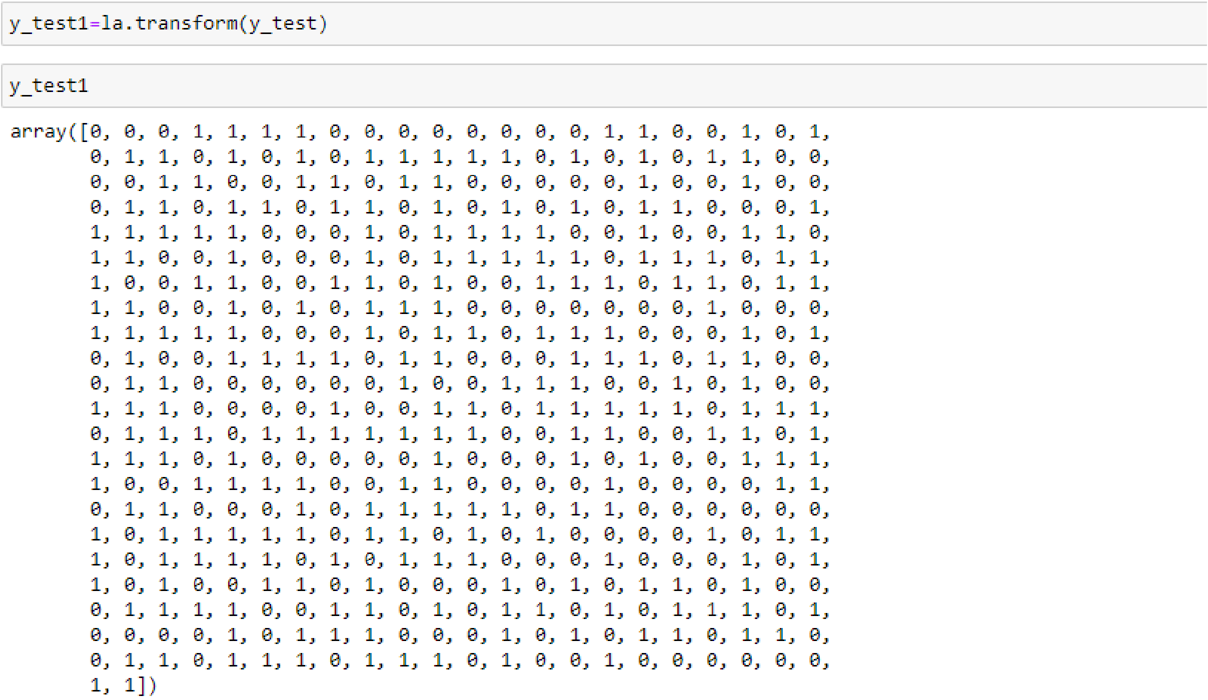
```
df.columns
```

```
Index(['step', 'type', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
       'oldbalanceDest', 'newbalanceDest', 'isFraud'],
      dtype='object')
```

```
from sklearn.preprocessing import LabelEncoder

la = LabelEncoder()
y_train1 = la.fit_transform(y_train)
```

```
y_test1=la.transform(y_test)
```

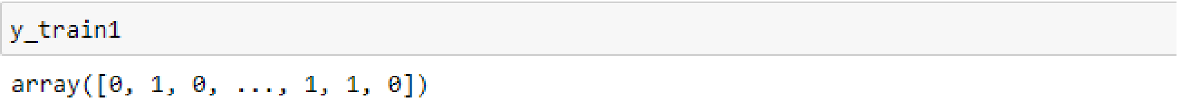
preprocessing class of sklearn. LabelEncoder[source] 0 to n classes-1 as the range for the target labels to be encoded. Instead of encoding the input X, the target values, i.e. y, should be encoded using this transformer.



```
y_test1=la.transform(y_test)
```

```
y_test1
```

```
array([0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1])
```

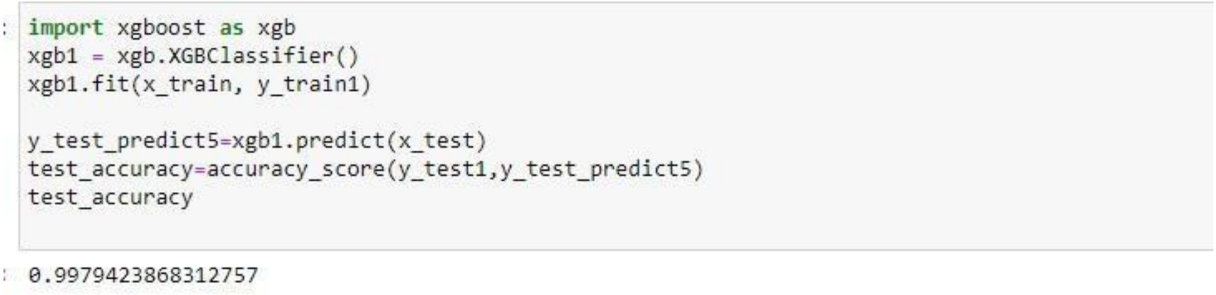


```
y_train1
```

```
array([0, 1, 0, ..., 1, 1, 0])
```

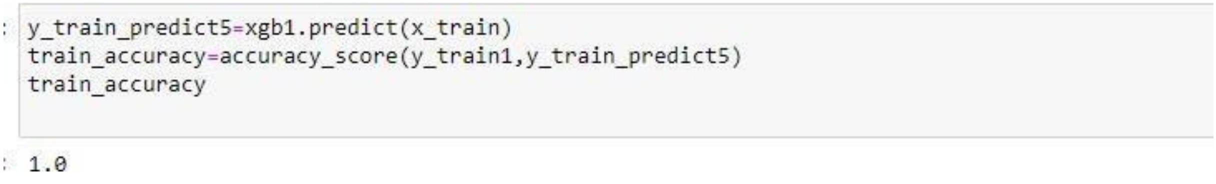
Xgboost Classifier

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, the xgboostClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, confusion matrix and classification report is done



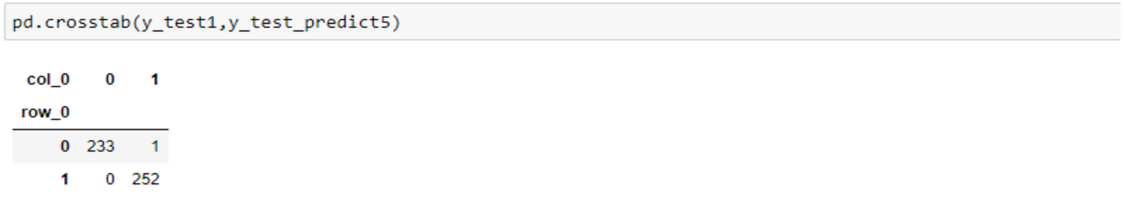
```
import xgboost as xgb
xgb1 = xgb.XGBClassifier()
xgb1.fit(x_train, y_train1)

y_test_predict5=xgb1.predict(x_test)
test_accuracy=accuracy_score(y_test1,y_test_predict5)
test_accuracy
```

```
0.9979423868312757
```



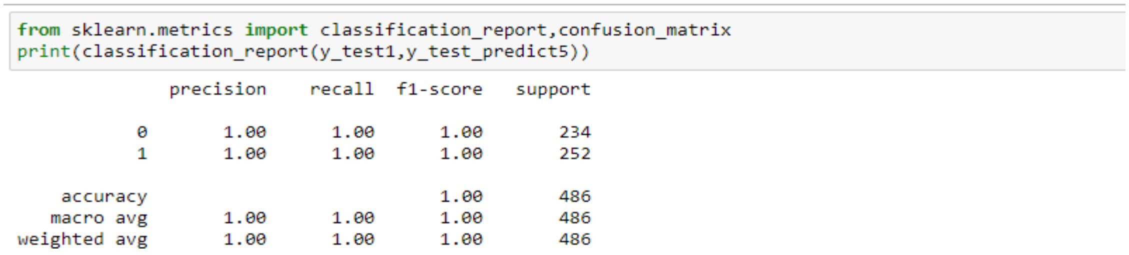
```
y_train_predict5=xgb1.predict(x_train)
train_accuracy=accuracy_score(y_train1,y_train_predict5)
train_accuracy
```

```
1.0
```



```
pd.crosstab(y_test1,y_test_predict5)
```

col_0	0	1
row_0		
0	233	1
1	0	252



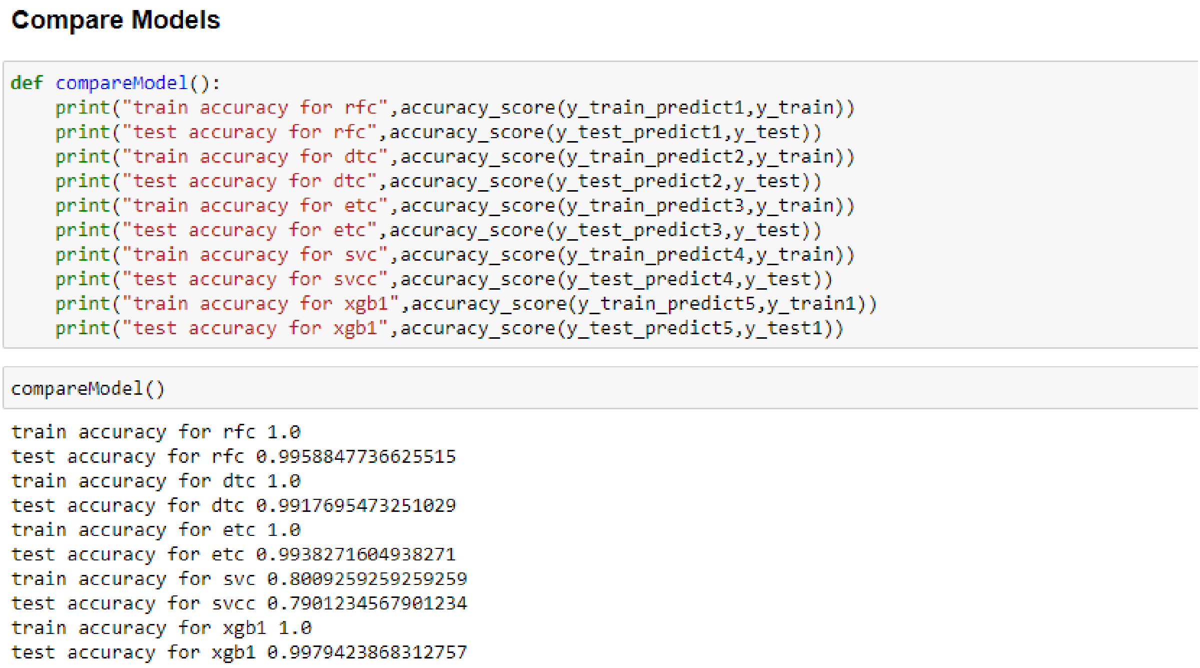
```
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test1,y_test_predict5))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       234
           1       1.00      1.00      1.00       252

    accuracy                           1.00       486
   macro avg       1.00      1.00      1.00       486
weighted avg       1.00      1.00      1.00       486
```

Compare The Models

For comparing the above four models, the compareModel function is defined.

After calling the function, the results of models are displayed as output. From the five models, the svc is performing well. From the below image, We can see the accuracy of the model is 79% accuracy. .



Compare Models

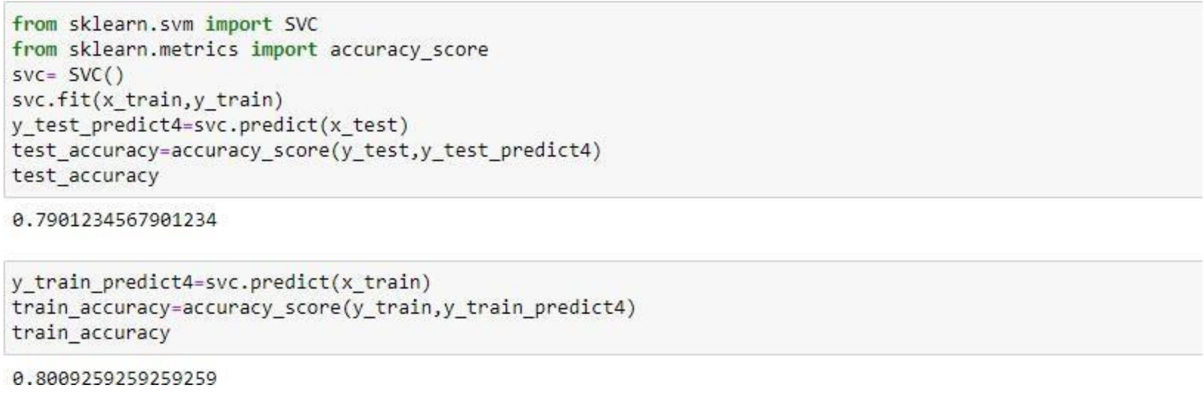
```
def compareModel():
    print("train accuracy for rfc",accuracy_score(y_train_predict1,y_train))
    print("test accuracy for rfc",accuracy_score(y_test_predict1,y_test))
    print("train accuracy for dtc",accuracy_score(y_train_predict2,y_train))
    print("test accuracy for dtc",accuracy_score(y_test_predict2,y_test))
    print("train accuracy for etc",accuracy_score(y_train_predict3,y_train))
    print("test accuracy for etc",accuracy_score(y_test_predict3,y_test))
    print("train accuracy for svc",accuracy_score(y_train_predict4,y_train))
    print("test accuracy for svcc",accuracy_score(y_test_predict4,y_test))
    print("train accuracy for xgb1",accuracy_score(y_train_predict5,y_train1))
    print("test accuracy for xgb1",accuracy_score(y_test_predict5,y_test1))
```

```
compareModel()
```

```
train accuracy for rfc 1.0
test accuracy for rfc 0.9958847736625515
train accuracy for dtc 1.0
test accuracy for dtc 0.9917695473251029
train accuracy for etc 1.0
test accuracy for etc 0.9938271604938271
train accuracy for svc 0.8009259259259259
test accuracy for svcc 0.7901234567901234
train accuracy for xgb1 1.0
test accuracy for xgb1 0.9979423868312757
```

Evaluating Performance Of The Model And Saving The Model

From sklearn, accuracy_score is used to evaluate the score of the model. On the parameters, we have given svc (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model is svc by pickle.dump().

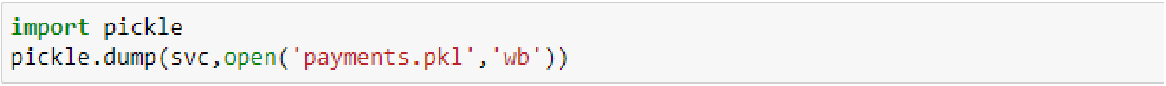


```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
svc= SVC()
svc.fit(x_train,y_train)
y_test_predict4=svc.predict(x_test)
test_accuracy=accuracy_score(y_test,y_test_predict4)
test_accuracy
```

```
0.7901234567901234
```

```
y_train_predict4=svc.predict(x_train)
train_accuracy=accuracy_score(y_train,y_train_predict4)
train_accuracy
```

```
0.8009259259259259
```



```
import pickle
pickle.dump(svc,open('payments.pkl','wb'))
```