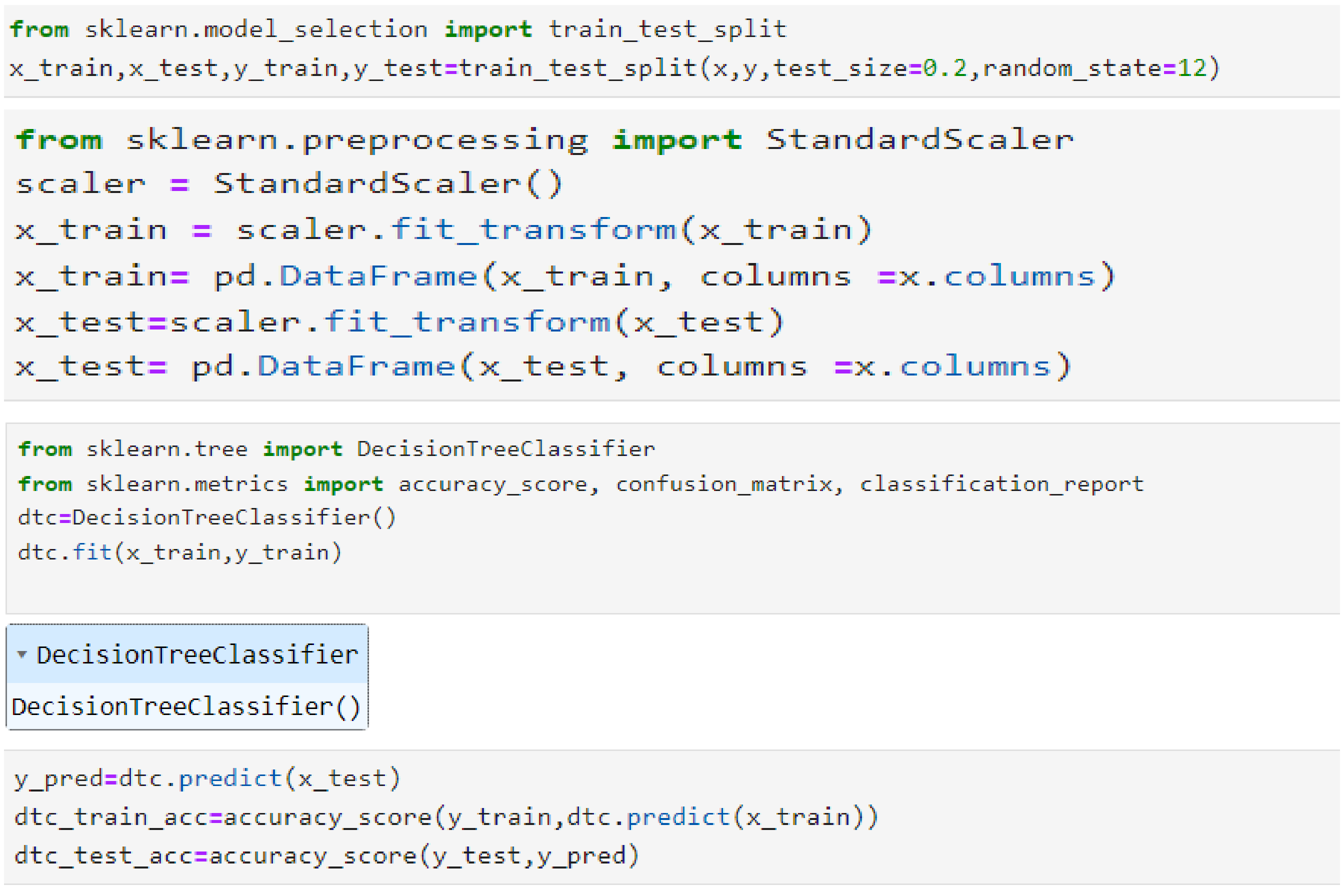
Model Development Phase Template

Date	18 June 2024
Team ID	739990
Project Title	Auto Insurance Fraud Detection
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:



```
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,random_state=12)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_train= pd.DataFrame(x_train, columns =x.columns)
x_test=scaler.fit_transform(x_test)
x_test= pd.DataFrame(x_test, columns =x.columns)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred=dtc.predict(x_test)
dtc_train_acc=accuracy_score(y_train,dtc.predict(x_train))
dtc_test_acc=accuracy_score(y_test,y_pred)
```

```
print(f"Training accuracy of Decision Tree is : {dtc_train_acc}")
print(f"Test accuracy of Decision Tree is : {dtc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
#RandomForest
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(criterion= 'entropy', max_depth= 10, max_features= 'sqrt', min_samples_leaf= 1, min_samples_split= 3, n_estimators= 140)
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
rand_clf_train_acc = accuracy_score(y_train, rfc.predict(x_train))
rand_clf_test_acc = accuracy_score(y_test, y_pred)
```

```
y_pred
```

```
print(f"Training accuracy of Random Forest is : {rand_clf_train_acc}")
print(f"Test accuracy of Random Forest is : {rand_clf_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
#KNN
```

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors = 30)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
knn_train_acc = accuracy_score(y_train, knn.predict(x_train))
knn_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of KNN is : {knn_train_acc}")
print(f"Test accuracy of KNN is : {knn_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
#Logistic Regression
```

```
from sklearn.linear_model import LogisticRegression,LogisticRegressionCV
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

```
lg=LogisticRegressionCV(solver='lbfgs',max_iter=5000,cv=10)
```

```
lg.fit(x_train,y_train)
y_pred=lg.predict(x_test)
```

```
lg_train_acc = accuracy_score(y_train, lg.predict(x_train))
lg_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of LogisticRegression is : {lg_train_acc}")
print(f"Test accuracy of LogisticRegression is : {lg_test_acc}")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
#Naive Bayes
```

```
from sklearn.naive_bayes import CategoricalNB,GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

```
gnb=GaussianNB()
```

```
model=gnb.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

```
gnb_train_acc = accuracy_score(y_train, gnb.predict(x_train))
gnb_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of NaiveBayes is : {gnb_train_acc}")
print(f"Test accuracy of NaiveBayes is : {gnb_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
#Support Vector Classifier
```

```
from sklearn.svm import SVC
svc=SVC()
svc.fit(x_train,y_train)
y_pred=svc.predict(x_test)
```

```
svc_train_acc = accuracy_score(y_train, svc.predict(x_train))
svc_test_acc = accuracy_score(y_test, y_pred)

print(f"Training accuracy of  is Support Vector Machine: {svc_train_acc}")
print(f"Test accuracy of Support Vector machine is : {svc_test_acc}")

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix

Decision Tree	`1.0`	68%	`[[88 64]`
			` [20 28]]`

Random Forest	`0.9900166389351082`	**64%**	`[[83 69]`
			` [25 23]]`

KNN	`0.6414309484193012`	**37%**	`[[ 36 116]`
			` [  7  41]]`

Logistic Regression	`0.6913477537437605`	**64%**	`[[82 70]`
			` [23 25]]`

Naive Bayes	`0.6921797004991681`	**54%**	`[[59 93]`
			` [17 31]]`

Support Vector Classifier	`0.8926788685524126`	**72%**	`[[101  51]`
			` [ 27  21]]`