Model Development Phase Template

Date	20-Jun-25
Team ID	SWTID1751617613
Project Title	Auto Fraud Detection
Marks	5 marks

Model Training code and Evaluation

Code and Output

In [ ]:
```
# Split into 80% train and 20% test (you can change test_size if needed)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42,
                                                    stratify=y)
```

In [ ]:
```
from imblearn.over_sampling import SMOTE

# Create SMOTE object
smt = SMOTE(random_state=42)
# Apply SMOTE only on training data
X_train_resampled, y_train_resampled = smt.fit_resample(X_train, y_train)
```

In [ ]:
```
std_scaler = StandardScaler()

X_train = std_scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train, columns=X.columns)

X_test = std_scaler.transform(X_test)
X_test = pd.DataFrame(X_test, columns=X.columns)
```

Decision Tree

In [ ]:
```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
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)

# You can print the accuracies to see the results
print(f"Decision Tree Training Accuracy: {dtc_train_acc:.4f}")
print(f"Decision Tree Test Accuracy: {dtc_test_acc:.4f}")
```

```
Decision Tree Training Accuracy: 1.0000
Decision Tree Test Accuracy: 0.7800
```

Random Forest Classifier

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score # You'll need to import this

# Assuming X_train, y_train, X_test, y_test are already defined and loaded

rfc = RandomForestClassifier(criterion='entropy', max_depth=10, max_features='sqrt', min_samples_leaf=1, min_samples_spl
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

rfc_train_acc = accuracy_score(y_train, rfc.predict(X_train))
rfc_test_acc = accuracy_score(y_test, y_pred)

print(f"Random Forest Training Accuracy: {rfc_train_acc:.4f}")
print(f"Random Forest Test Accuracy: {rfc_test_acc:.4f}")
```

```
Random Forest Training Accuracy: 0.9975
Random Forest Test Accuracy: 0.7750
```

KNN Model

In [ ]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report # You'll need to import these

# Assuming X_train, y_train, X_test, y_test are already defined and loaded

knn = KNeighborsClassifier(n_neighbors=30)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

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

```
[[151   0]
 [ 48   1]]
              precision    recall  f1-score   support

           0       0.76      1.00      0.86       151
           1       1.00      0.02      0.04        49

    accuracy                           0.76       200
   macro avg       0.88      0.51      0.45       200
weighted avg       0.82      0.76      0.66       200
```

Logistic Regression

In [ ]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# Assuming X_train, y_train, X_test, y_test are already defined and loaded

lg = LogisticRegressionCV(solver='lbfgs', max_iter=5000, cv=10)
lg.fit(X_train, y_train)

lrg_pred = lg.predict(X_test)

print("Confusion Matrix:")
print(confusion_matrix(y_test, lrg_pred))

# Calculate and print training accuracy
lrg_train_acc = accuracy_score(y_train, lg.predict(X_train))
print(f"\nLogistic Regression Training Accuracy: {lrg_train_acc:.4f}")

# Calculate and print test accuracy
lrg_test_acc = accuracy_score(y_test, lrg_pred)
print(f"Logistic Regression Test Accuracy: {lrg_test_acc:.4f}")

# You might also want to print the classification report for more detailed metrics
# print("\nClassification Report:")
# print(classification_report(y_test, lrg_pred))
```

```
Confusion Matrix:
[[142   9]
 [ 30  19]]

Logistic Regression Training Accuracy: 0.7850
Logistic Regression Test Accuracy: 0.8050
```

Guassian Naïve Bayes

In [ ]:

```
from sklearn.naive_bayes import CategoricalNB, GaussianNB
from sklearn.metrics import accuracy_score # Import accuracy_score

# Assuming X_train, y_train, X_test, y_test are already defined and loaded

gnb = GaussianNB()
model_2 = gnb.fit(X_train, y_train) # Training the Gaussian Naive Bayes model
predict_log = model_2.predict(X_test) # Making predictions on the test set

# Print Training Accuracy
# Note: The original image had '100*' for percentage, which is common for presentation.
# If you want the raw accuracy score (0 to 1), remove '*100'.
print("Training Accuracy", 100 * accuracy_score(model_2.predict(X_train), y_train))

# Print Testing Accuracy
print("Testing Accuracy", 100 * accuracy_score(y_test, predict_log))
```

```
Training Accuracy 71.875
Testing Accuracy 72.0
```

SVC

In [ ]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report # Import necessary metrics

# Assuming X_train, y_train, X_test, y_test are already defined and loaded

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 SVC : {svc_train_acc}")
print(f"Test accuracy of SVC : {svc_test_acc}")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Training accuracy of SVC : 0.84375
Test accuracy of SVC : 0.755
[[150   1]
 [ 48   1]]
              precision    recall  f1-score   support

           0       0.76      0.99      0.86       151
           1       0.50      0.02      0.04        49

    accuracy                           0.76       200
   macro avg       0.63      0.51      0.45       200
weighted avg       0.69      0.76      0.66       200
```