```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsOneClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

```
df = pd.read_csv("/content/synthetic_FINANCE.csv")
```

```
df.head()
```

	Unnamed: 0	Age_of_Account_years	Number_of_Transactions_last_month	Average_Transaction_Value	Credit_Score	Account_Balance	Risk_Cla
0	0	7	12	5577.48	383	3804.62	
1	1	20	55	8607.71	751	11394.94	
2	2	29	13	5355.35	684	36539.28	
3	3	15	23	1852.78	471	29980.53	

```
df.describe()
```

	Unnamed: 0	Age_of_Account_years	Number_of_Transactions_last_month	Average_Transaction_Value	Credit_Score	Account_Balance	Risl
count	700.00000	700.000000	700.000000	700.000000	700.000000	700.000000	700
mean	349.50000	15.204286	48.458571	5151.383714	582.905714	25794.607714	1
std	202.21688	8.852292	28.668841	2858.902788	157.491457	14466.345341	0
min	0.00000	1.000000	1.000000	12.370000	300.000000	122.090000	0
25%	174.75000	7.000000	24.000000	2735.220000	448.000000	13428.705000	0
50%	349.50000	16.000000	49.000000	5291.260000	576.000000	26592.475000	1
75%	524.25000	23.000000	73.000000	7587.947500	722.250000	38184.982500	2

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 7 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Unnamed: 0                         700 non-null    int64  
 1   Age_of_Account_years               700 non-null    int64  
 2   Number_of_Transactions_last_month  700 non-null    int64  
 3   Average_Transaction_Value          700 non-null    float64
 4   Credit_Score                       700 non-null    int64  
 5   Account_Balance                    700 non-null    float64
 6   Risk_Class                         700 non-null    int64  
dtypes: float64(2), int64(5)
memory usage: 38.4 KB
```

```
# Display the current column names
print(df.columns)
```

```
Index(['Unnamed: 0', 'Age_of_Account_years',
       'Number_of_Transactions_last_month', 'Average_Transaction_Value',
       'Credit_Score', 'Account_Balance', 'Risk_Class'],
      dtype='object')
```

```
#removing the unamed column
df = df.drop("Unnamed: 0", axis=1)

X = df.drop("Risk_Class", axis=1)
y = df["Risk_Class"]
```

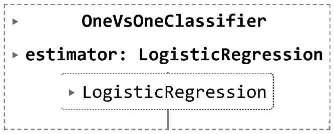
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 6 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Age_of_Account_years              700 non-null    int64  
 1   Number_of_Transactions_last_month 700 non-null    int64  
 2   Average_Transaction_Value         700 non-null    float64
 3   Credit_Score                      700 non-null    int64  
 4   Account_Balance                   700 non-null    float64
 5   Risk_Class                        700 non-null    int64  
dtypes: float64(2), int64(4)
memory usage: 32.9 KB
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
num_of_classifiers = len(df["Risk_Class"].unique())
```

```
model = OneVsOneClassifier(LogisticRegression())
model.fit(X_train, y_train)
```



▸ **OneVsOneClassifier**

▸ **estimator: LogisticRegression**

▸ LogisticRegression

```
y_pred = model.predict(X_test)
```

```
confusion_matrix = confusion_matrix(y_test, y_pred)
print("The Confusion Matrix is:\n", confusion_matrix)
```

```
The Confusion Matrix is:
 [[34  3 12]
 [10  8 15]
 [13  9 36]]
```

```
class_report = classification_report(y_test, y_pred)
print("The Classification Report:\n", class_report)
```

```
The Classification Report:
               precision    recall  f1-score   support

           0       0.60      0.69      0.64        49
           1       0.40      0.24      0.30        33
           2       0.57      0.62      0.60        58

    accuracy                           0.56       140
   macro avg       0.52      0.52      0.51       140
weighted avg       0.54      0.56      0.54       140
```

The classification report shows the result that our model has an overall accuracy of 56%.

```
for i in range(num_of_classifiers):
    for j in range(i + 1, num_of_classifiers):
        class_i_vs_j = f"{i}_vs_{j}"
        predictions_i_vs_j = (y_pred == class_i_vs_j)
        print(f"Predictions for {class_i_vs_j}: {sum(predictions_i_vs_j)}")
```

```
Predictions for 0_vs_1: 0
Predictions for 0_vs_2: 0
Predictions for 1_vs_2: 0
```