```
# Data Manipulation and Linear Algebra
import pandas as pd
import numpy as np

# Plots
import seaborn as sns
sns.set_style("darkgrid")
import matplotlib.pyplot as plt

# Machine Learning
from sklearn.model_selection import StratifiedShuffleSplit, cross_val_score, cross_val_predic
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, precisio
from sklearn.decomposition import PCA

from sklearn import tree, linear_model, ensemble

#ignore warning messages
import warnings
warnings.filterwarnings('ignore')


data = pd.read_csv("https://datahub.io/machine-learning/creditcard/r/creditcard.csv")
data
```

	Time	V1	V2	V3	V4	V5	V6	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.23959
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.07880
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.79146
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.23760
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.59294
...	...	...	...	...	...	...	...	.
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.91821
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.02433
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.29682
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.68618
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.57700

284807 rows × 31 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  object 
dtypes: float64(30), object(1)
memory usage: 67.4+ MB
```

```
data.drop("Time", axis=1, inplace=True)
```

```
data
```

	V1	V2	V3	V4	V5	V6	V7	V
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0986
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0851
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2476
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3774
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2705
...	...	...	...	...	...	...	...	
284802	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.3053

```
# Reducing the Number of Features in the Dataset using PCA
pca = PCA(2)
pca_dataframe = pd.DataFrame(pca.fit_transform(data.iloc[:, :29]), columns=["PCA1", "PCA2"])
```

284805	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.6791

```
full_data = pd.concat([pca_dataframe, data.iloc[:, 29:]], axis=1)
full_data
```

	PCA1	PCA2	Class
0	61.271382	1.319313	'0'
1	-85.661826	-1.043741	'0'
2	290.316696	0.810795	'0'
3	35.151659	0.928345	'0'
4	-18.360281	1.317343	'0'
...	...	...	...
284802	-87.586281	13.128921	'0'
284803	-63.560584	0.876761	'0'
284804	-20.470739	-1.970752	'0'
284805	-78.350638	0.408096	'0'
284806	128.652188	0.358654	'0'

284807 rows × 3 columns

```
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(full_data, full_data['Class']):
    train = full_data.loc[train_index]
    test = full_data.loc[test_index]


X_train = train.drop("Class", axis=1)
```

```
y_train = train["Class"]

X_test = test.drop("Class", axis=1)
y_test = test["Class"]

MLA_compare = pd.DataFrame()

row_index = 0

def MLA_testing(MLA, X_train, X_test, y_train, y_test):
    global row_index

    # Training The Model
    MLA.fit(X_train, y_train)

    # KFold Accuracies on Training Data
    kfold_accuracy = cross_val_score(estimator = MLA, X = X_train, y = y_train, cv = 10, n_jo
    print("K-Fold Accuracies:\n", kfold_accuracy, "\n")

    # Prediction on Testing Data
    y_pred = cross_val_predict(estimator = MLA, X = X_test, y = y_test, cv = 10, n_jobs=-1)

    # Accuracy for y_test and y_pred
    classifier_accuracy_score = accuracy_score(y_test, y_pred)
    print("Accuracy Score:\n", classifier_accuracy_score, "\n")

    # Confusion Matrix
    conf_mtx = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:\n", conf_mtx, "\n")

    # Classification Report
    class_rep = classification_report(y_test, y_pred)
    print("Classification Report:\n", class_rep, "\n")

    # Precision - Recall Curve
    yhat = MLA.predict_proba(X_test)
    no_skill = len(full_data.Class[full_data.Class==1]) / len(full_data.Class)
    precision, recall, _ = precision_recall_curve(y_test, yhat[:, 1])

    plt.figure(dpi=100, figsize=(15, 6))
    plt.subplot(121)
    sns.lineplot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
    sns.lineplot(recall, precision, marker='.', label=MLA.__class__.__name__)
    plt.title("Recall vs Precision Curve")
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.legend()

     # ROC Curve
    plt.subplot(122)
    sns.lineplot([0, 1], [0, 1], linestyle='--', label='No Skill')
    fpr, tpr, _ = roc_curve(y_test, yhat[:, 1])
```

```
        fpr, tpr, _ = roc_curve(y_test, yhat[:, 1])
        sns.lineplot(fpr, tpr, marker='.', label=MLA.__class__.__name__)
        plt.title("ROC Curve")
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.legend()
        plt.show()

        # Saving Data in Dataframe
        MLA_name = MLA.__class__.__name__
        MLA_compare.loc[row_index, 'MLA Name'] = MLA_name
        MLA_compare.loc[row_index, 'Accuracy Score'] = classifier_accuracy_score*100
        MLA_compare.loc[row_index, 'K-Fold Accuracy'] = kfold_accuracy.mean()*100

        print(MLA_name, "Done")

        row_index+=1


    rf_clf = ensemble.RandomForestClassifier()

    MLA_testing(rf_clf, X_train, X_test, y_train, y_test)
```

```
K-Fold Accuracies:
 [0.9984639  0.99863946 0.99824446 0.99828835 0.99837612 0.99850772
 0.99859551 0.99859551 0.99841994 0.99837605]

Accuracy Score:
 0.9982971103542713

Confusion Matrix:
 [[56857     7]
 [   90   811]
```

```
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```