```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
!file /content/creditcard.csv.zip
```

```
/content/creditcard.csv.zip: cannot open `/content/creditcard.csv.zip' (No such file or directory)
```

```
df = pd.read_csv('/content/creditcard.csv')
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270

5 rows × 31 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21878 entries, 0 to 21877
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Time    21878 non-null  int64
 1   V1      21878 non-null  float64
 2   V2      21878 non-null  float64
 3   V3      21878 non-null  float64
 4   V4      21878 non-null  float64
 5   V5      21878 non-null  float64
 6   V6      21878 non-null  float64
 7   V7      21878 non-null  float64
 8   V8      21878 non-null  float64
 9   V9      21878 non-null  float64
 10  V10     21878 non-null  float64
 11  V11     21878 non-null  float64
 12  V12     21878 non-null  float64
 13  V13     21878 non-null  float64
 14  V14     21878 non-null  float64
 15  V15     21878 non-null  float64
 16  V16     21878 non-null  float64
 17  V17     21878 non-null  float64
 18  V18     21878 non-null  float64
 19  V19     21878 non-null  float64
 20  V20     21878 non-null  float64
 21  V21     21878 non-null  float64
 22  V22     21878 non-null  float64
 23  V23     21878 non-null  float64
 24  V24     21878 non-null  float64
 25  V25     21878 non-null  float64
 26  V26     21878 non-null  float64
 27  V27     21878 non-null  float64
 28  V28     21878 non-null  float64
 29  Amount  21877 non-null  float64
 30  Class   21877 non-null  float64
dtypes: float64(30), int64(1)
memory usage: 5.2 MB
```

```
print('Shape Of The Dataset', df.shape)
```

```
print('Class Categories', df['Class'].unique())
```

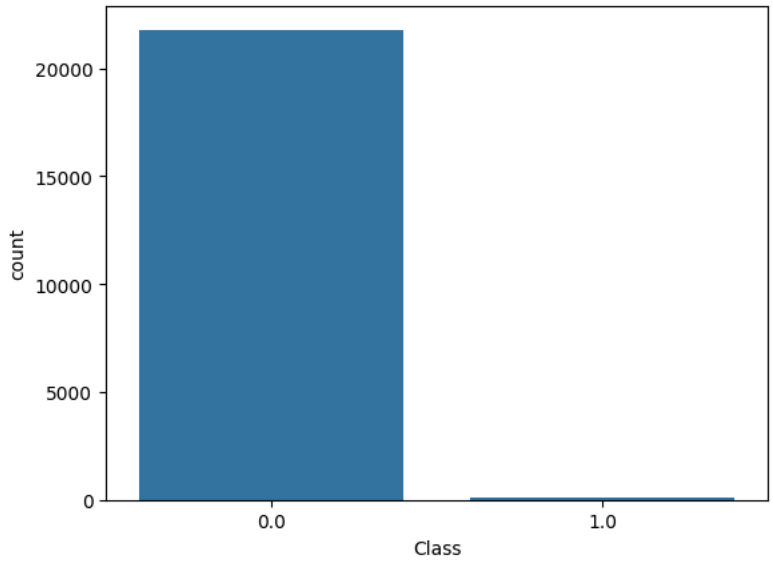
```
print('Number Of Records With The Class Value 0: ', (df.Class == 0).sum())
```

```
print('Number Of Records With The Class Value 1: ', (df.Class == 1).sum())
```

```
Shape Of The Dataset (21878, 31)
Class Categories [ 0.  1. nan]
Number Of Records With The Class Value 0:  21791
Number Of Records With The Class Value 1:  86
```

```
sns.countplot(x='Class', data=df)
```

```
<Axes: xlabel='Class', ylabel='count'>
```



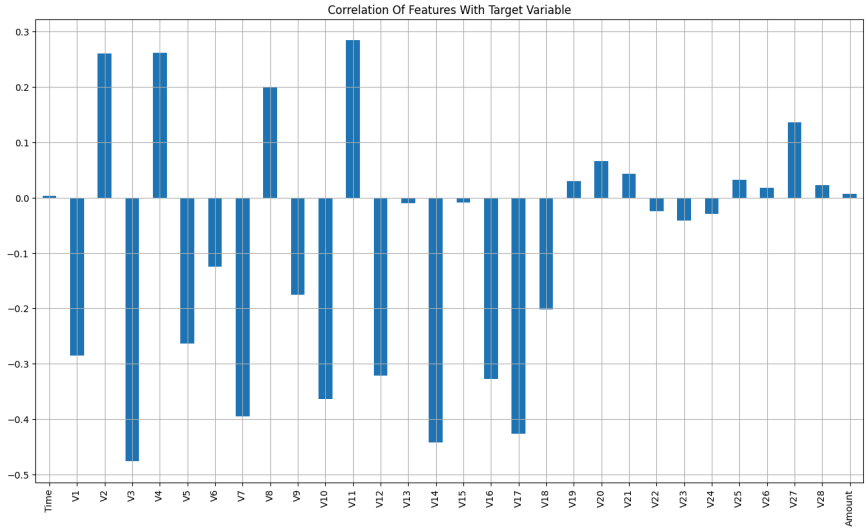

```
x = df.corr()['Class'][:30]
x
```

```
Time      0.003817
V1       -0.285602
V2        0.259959
V3       -0.476294
V4        0.261522
V5       -0.263605
V6       -0.124538
V7       -0.395674
V8        0.200598
V9       -0.175465
V10      -0.363563
V11       0.284211
V12      -0.321828
V13      -0.009892
V14      -0.441833
V15      -0.009043
V16      -0.327048
V17      -0.426317
V18      -0.201947
V19       0.029348
V20       0.065875
V21       0.042885
V22      -0.024029
V23      -0.041799
V24      -0.029307
V25       0.032147
V26       0.017745
V27       0.135965
V28       0.022922
Amount    0.006824
Name: Class, dtype: float64
```

```
x = df.corr()['Class'][:30]

x.plot.bar(figsize=(16, 9), title="Correlation Of Features With Target Variable", grid=True)
```

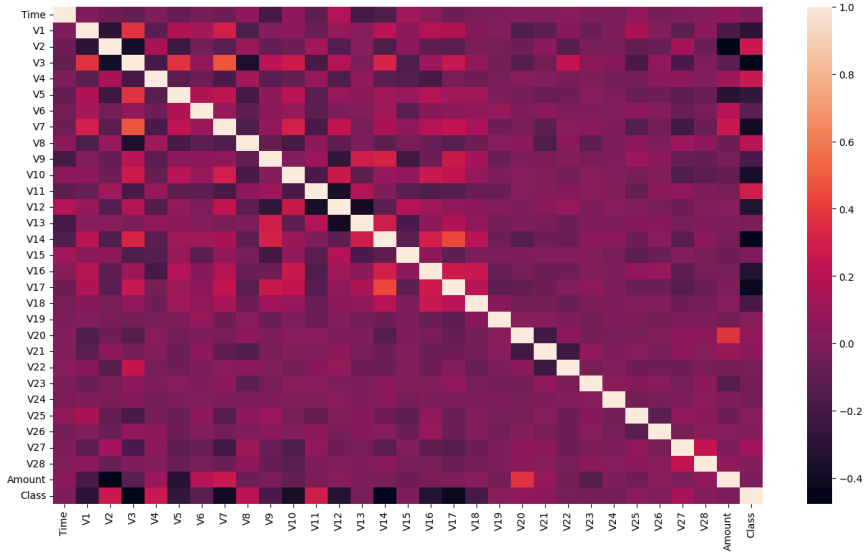
```
<Axes: title={'center': 'Correlation Of Features With Target Variable'}>
```




```
plt.figure(figsize=(16, 9))

sns.heatmap(df.corr())
```

```
<Axes: >
```




```
y = df.corr()['Class']

df2 = df.copy()

for i in df.columns:
    if abs(y[i]) < 0.13:
        df2.drop(columns=[i], inplace=True)


df2.head()
```

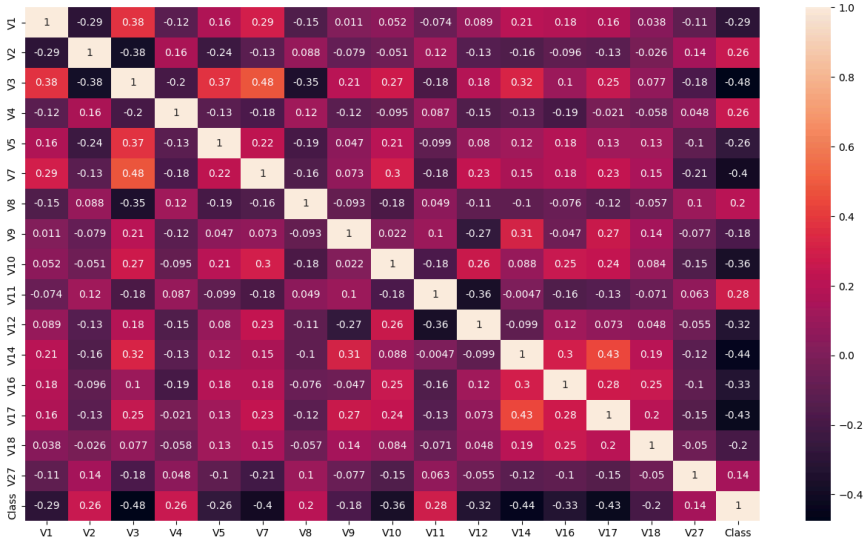
	V3	V4	V10	V11	V12	V14	V16	V17	C
0	2.536347	1.378155	0.090794	-0.551600	-0.617801	-0.311169	-0.470401	0.207971	
1	0.166480	0.448154	-0.166974	1.612727	1.065235	-0.143772	0.463917	-0.114805	
2	1.773209	0.379780	0.207643	0.624501	0.066084	-0.165946	-2.890083	1.109969	
3	1.792993	-0.863291	-0.054952	-0.226487	0.178228	-0.287924	-1.059647	-0.684093	
4	1.548718	0.403034	0.753074	-0.822843	0.538196	-1.119670	-0.451449	-0.237033	

Next steps: Generate code with df2 View recommended plots

```
plt.figure(figsize=(16, 9))

sns.heatmap(df2.corr(), annot=True)
```

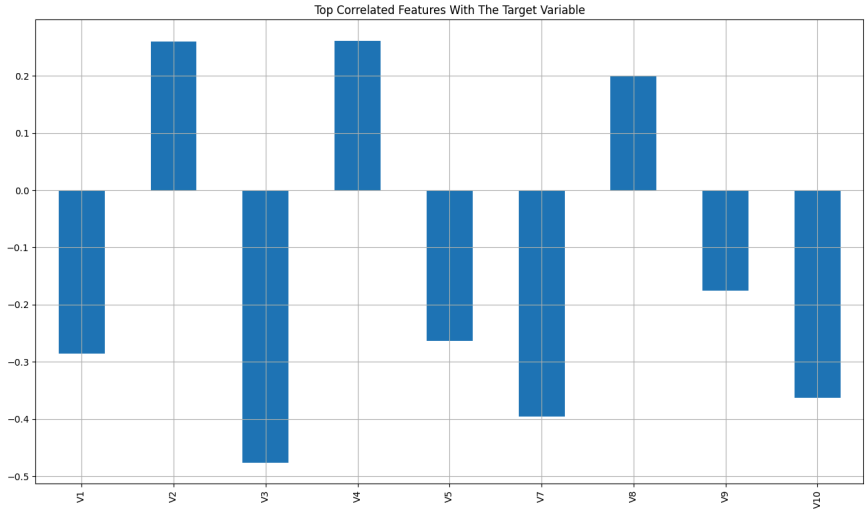
<Axes: >




```
x = df2.corr()['Class'][:9]

x.plot.bar(figsize=(16, 9), title="Top Correlated Features With The Target Variable", grid=True)
```

```
<Axes: title={'center': 'Top Correlated Features With The Target Variable'}>
```




```
print(y.isnull().sum())

df2.dropna(subset=['Class'], inplace=True)
X = df2.drop('Class', axis=1)
y = df2['Class']

rus = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = rus.fit_resample(X, y)

# ... rest of your code
```

```
0
```

```
from imblearn.under_sampling import RandomUnderSampler

# Separate features (X) and target (y)
X = df2.drop('Class', axis=1)
y = df2['Class']

# Initialize RandomUnderSampler
rus = RandomUnderSampler(random_state=42)

# Fit and apply the resampler to the data
X_resampled, y_resampled = rus.fit_resample(X, y)

# Convert the resampled data back to a DataFrame
downsampled_df = pd.concat([pd.DataFrame(X_resampled, columns=X.columns), pd.DataFrame(y_resampled, columns=['Class'])], axis=1)


downsampled_df.head()
```

	V1	V2	V3	V4	V5	V7	V8	V9
0	1.080471	-0.898196	-0.101167	-2.207604	-0.807356	-0.345683	0.071073	1.866907
1	-1.642711	-0.987960	2.219241	-3.413385	0.231503	-0.360119	0.472436	3.187289
2	-2.138696	1.377871	1.642452	-0.013499	-0.109421	-0.633353	-3.075967	0.554068
3	-1.055962	0.843840	1.568174	-0.011194	0.422437	0.566527	0.422157	-0.068220
4	-0.334535	0.241851	1.727856	-0.862383	-0.580904	-0.040325	-0.181736	0.383812

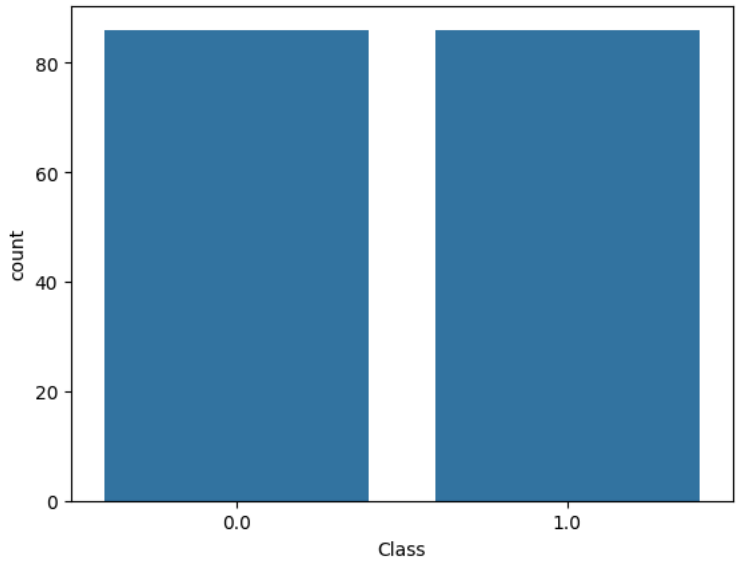
Next steps: Generate code with downsampled_df View recommended plots

```
downsampled_df.shape
```

```
(172, 17)
```

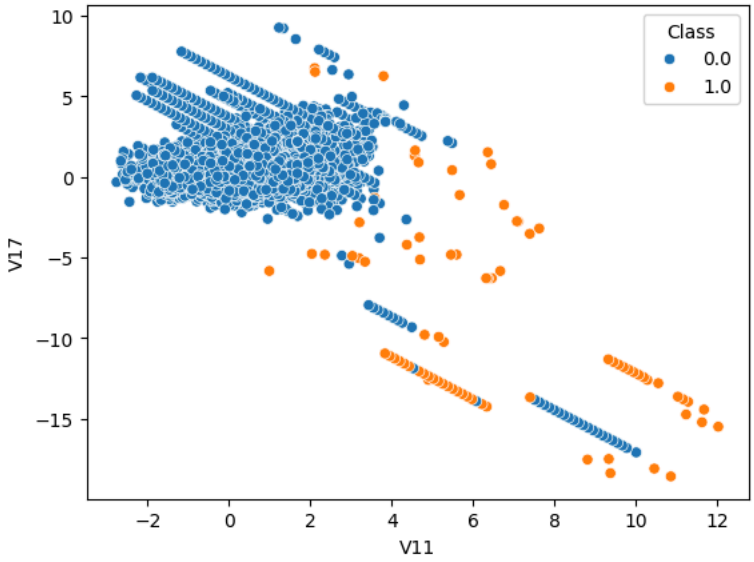
```
sns.countplot(x='Class', data=downsampled_df)
```

```
<Axes: xlabel='Class', ylabel='count'>
```




```
sns.scatterplot(x='V11', y='V17', hue='Class', data=df2)
```

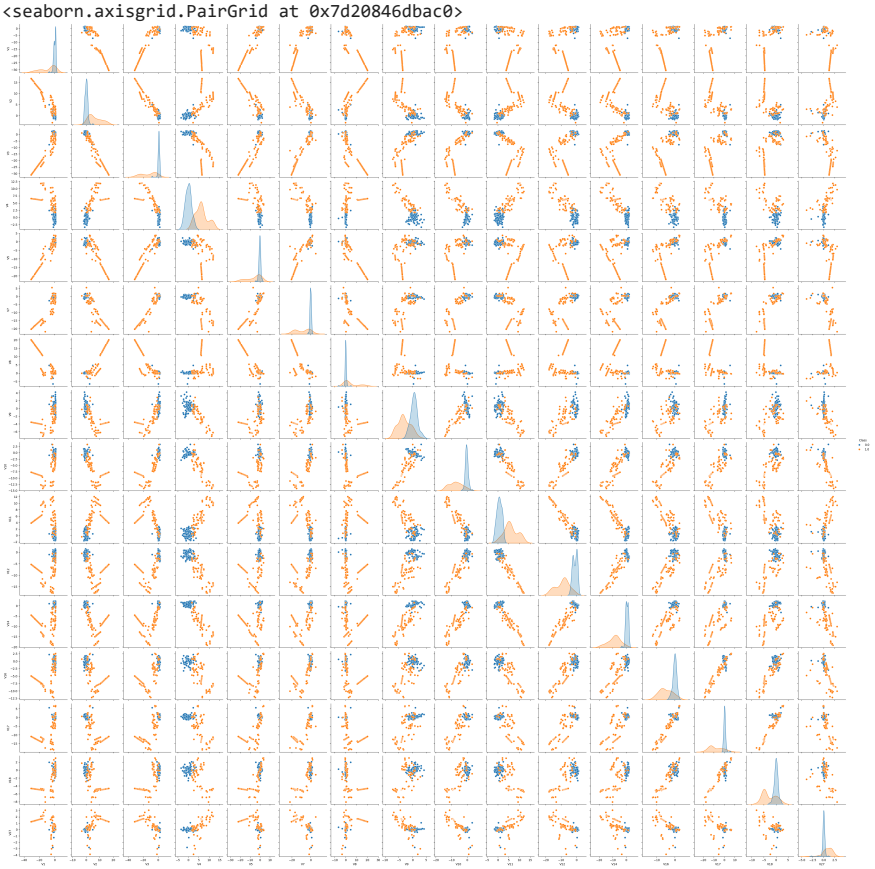
```
<Axes: xlabel='V11', ylabel='V17'>
```




```
import warnings

# To ignore all warnings
warnings.filterwarnings('ignore')
# Pair Plot of all variables
sns.pairplot(downsampled_df, hue='Class')
```

```
<seaborn.axisgrid.PairGrid at 0x7d20846dbac0>
```



```
!pip install lazypredict
```

```
Collecting lazypredict
  Downloading lazypredict-0.2.12-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from lazypredict) (8.1.7)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.66.4)
```

```
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.4.2)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.1.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.11.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->lazypredict) (3.5
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->lazypredict
Installing collected packages: lazypredict
Successfully installed lazypredict-0.2.12
```

```
from sklearn.model_selection import train_test_split
from lazypredict.Supervised import LazyClassifier

# Separate features and target
x = downsampled_df.drop(columns= 'Class')
y = downsampled_df['Class']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)


# Fit all models
clf = LazyClassifier(predictions=True)
models, predictions = clf.fit(x_train, x_test, y_train, y_test)
models
```

```
100%|██████████| 29/29 [00:01<00:00, 23.17it/s][LightGBM] [Info] Number of positive:
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 757
[LightGBM] [Info] Number of data points in the train set: 137, number of used feature
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.503650 -> initscore=0.014599
[LightGBM] [Info] Start training from score 0.014599
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Model					
AdaBoostClassifier	1.00	1.00	1.00	1.00	0.13
BaggingClassifier	1.00	1.00	1.00	1.00	0.04
XGBClassifier	1.00	1.00	1.00	1.00	0.09
DecisionTreeClassifier	1.00	1.00	1.00	1.00	0.02
RandomForestClassifier	1.00	1.00	1.00	1.00	0.18
QuadraticDiscriminantAnalysis	1.00	1.00	1.00	1.00	0.04
SGDClassifier	0.97	0.97	0.97	0.97	0.02
LGBMClassifier	0.97	0.97	0.97	0.97	0.15
PassiveAggressiveClassifier	0.97	0.97	0.97	0.97	0.02
ExtraTreesClassifier	0.97	0.97	0.97	0.97	0.14
KNeighborsClassifier	0.97	0.97	0.97	0.97	0.02
LinearSVC	0.97	0.97	0.97	0.97	0.02
LabelSpreading	0.94	0.94	0.94	0.94	0.02
LabelPropagation	0.94	0.94	0.94	0.94	0.02
LinearDiscriminantAnalysis	0.94	0.94	0.94	0.94	0.04
LogisticRegression	0.94	0.94	0.94	0.94	0.03
NuSVC	0.94	0.94	0.94	0.94	0.02
Perceptron	0.94	0.94	0.94	0.94	0.02
RidgeClassifier	0.94	0.94	0.94	0.94	0.03
RidgeClassifierCV	0.94	0.94	0.94	0.94	0.02
CalibratedClassifierCV	0.94	0.94	0.94	0.94	0.05
SVC	0.94	0.94	0.94	0.94	0.02
GaussianNB	0.94	0.94	0.94	0.94	0.02
ExtraTreeClassifier	0.89	0.88	0.88	0.89	0.02
BernoulliNB	0.89	0.88	0.88	0.88	0.02
NearestCentroid	0.86	0.85	0.85	0.85	0.02

Next steps: Generate code with `models` View recommended plots

```
from sklearn.semi_supervised import LabelSpreading
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

X = downsampled_df.drop('Class', axis=1)
y = downsampled_df['Class']

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

model = LabelSpreading()

param_grid = {
    'kernel': ['knn', 'rbf'],
    'gamma': ['scale', 'auto', 0.1, 1.0],
    'alpha': [0.1, 0.2, 0.5, 0.8],
    'n_neighbors': [3, 5, 7]
}

grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy', verbose=1)

grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_

print(f"Best Hyperparameters: {best_params}")
print(f"Best Cross-validation Accuracy: {best_score:.2f}")

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print("\nClassification Report on Test Set:")
print(classification_report(y_test, y_pred))
```

```
Fitting 5 folds for each of 96 candidates, totalling 480 fits
Best Hyperparameters: {'alpha': 0.2, 'gamma': 1.0, 'kernel': 'rbf', 'n_neighbors': 3}
Best Cross-validation Accuracy: 0.96

Classification Report on Test Set:
              precision    recall  f1-score   support

         0.0       0.94      0.94      0.94        18
         1.0       0.94      0.94      0.94        17

    accuracy                           0.94        35
   macro avg       0.94      0.94      0.94        35
weighted avg       0.94      0.94      0.94        35
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.semi_supervised import LabelSpreading, LabelPropagation
from xgboost import XGBClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report, confusion_matrix

X = downsampled_df.drop(columns='Class')
y = downsampled_df['Class']


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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

label_spreading = LabelSpreading()
label_propagation = LabelPropagation()
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss')


ensemble = VotingClassifier(estimators=[
    ('label_spreading', label_spreading),
    ('label_propagation', label_propagation),
    ('xgb', xgb)
], voting='hard')

ensemble.fit(X_train, y_train)

y_pred = ensemble.predict(X_test)

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[[16  2]
 [ 0 17]]

Classification Report:
```

	precision	recall	f1-score	support
0.0	1.00	0.89	0.94	18
1.0	0.89	1.00	0.94	17
accuracy			0.94	35
macro avg	0.95	0.94	0.94	35
weighted avg	0.95	0.94	0.94	35

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.semi_supervised import LabelSpreading, LabelPropagation
from xgboost import XGBClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

```
X = downsampled_df.drop(columns='Class')
y = downsampled_df['Class']
```