Spotify Song Hit Prediction

Andrew Smith 12/5/2022

The purpose of this project is to try to predict if a song on Spotify (a popular music streaming service provider) will be a "Hit" or a "Flop" based on metrics derived from the Spotify API. The metrics rate the song along various properties such as "loudnes", "energy", "key", "temp", "duration", etc. The explaination and details on how the metrics are derived can be found at https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features

For this project, the data to be analyzed was obtained from Kaggle: https://www.kaggle.com/datasets/theoverman/the-spotify-hit-predictor-dataset

The curator of the data used the Spotify API to obtain the song metrics for several decades of songs and also added a "target" feature which labels each song as being a "Hit" or a "Flop". The curator made this determination based on other external sources which identified hit songs.

```
    - The track must not appear in the 'hit' list of that decade.
    - The track's artist must not appear in the 'hit' list of that
decade.
    - The track must belong to a genre that could be considered
non-mainstream and / or avant-garde.
    - The track's genre must not have a song in the 'hit' list.
    - The track must have 'US' as one of its markets.
```

We will split the data into a training and validation set, then run the training data through several different training models and see how well the fit is on the validation set.

In [1]:
```python
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
data\dataset-of-00s.csv
data\dataset-of-10s.csv
data\dataset-of-60s.csv
data\dataset-of-70s.csv
data\dataset-of-80s.csv
data\dataset-of-90s.csv
data\LICENSE
data\README.txt
```

Exploratory Data Analysis

First read in the datasets into a list of decade DataFrames

In [2]:
```
decade_list = [pd.read_csv(f'data/dataset-of-{decade}s.csv') for decade in ['60', '
```

Let's take a look at some of the data from the first decade.

In [3]:
```
decade_list[0].info()
decade_list[0].head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8642 entries, 0 to 8641
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   track             8642 non-null   object
 1   artist            8642 non-null   object
 2   uri               8642 non-null   object
 3   danceability      8642 non-null   float64
 4   energy            8642 non-null   float64
 5   key               8642 non-null   int64
 6   loudness          8642 non-null   float64
 7   mode              8642 non-null   int64
 8   speechiness       8642 non-null   float64
 9   acousticness      8642 non-null   float64
 10  instrumentalness  8642 non-null   float64
 11  liveness          8642 non-null   float64
 12  valence           8642 non-null   float64
 13  tempo             8642 non-null   float64
 14  duration_ms       8642 non-null   int64
 15  time_signature    8642 non-null   int64
 16  chorus_hit        8642 non-null   float64
 17  sections          8642 non-null   int64
 18  target            8642 non-null   int64
dtypes: float64(10), int64(6), object(3)
memory usage: 1.3+ MB
```

Out[3]:

	track	artist	uri	danceability	energy	key	loudness
0	Jealous Kind Of Fella	Garland Green	spotify:track:1dtKN6wwlolkM8XZy2y9C1	0.417	0.620	3	-7.727
1	Initials B.B.	Serge Gainsbourg	spotify:track:5hjsmSnUefdUqzsDogisiX	0.498	0.505	3	-12.475
2	Melody Twist	Lord Melody	spotify:track:6uk8tI6pwxxdVTNlNOJeJh	0.657	0.649	5	-13.392
3	Mi Bomba Sonó	Celia Cruz	spotify:track:7aNjMJ05FvUXACPWZ7yJmv	0.590	0.545	7	-12.058
4	Uravu Solla	P. Susheela	spotify:track:1rQ0clvgkzWr001POOPJWx	0.515	0.765	11	-3.515

Fold in the decade as feature and combine the decade DataFrames into one large DataFrame

In [4]:

```
for i, decade in enumerate([1960, 1970, 1980, 1990, 2000, 2010]):
    decade_list[i]['decade'] = pd.Series(decade, index=decade_list[i].index)

#data = pd.concat(decade_list, axis=0).sample(frac=0.2, random_state=1).reset_index
data = pd.concat(decade_list, axis=0).reset_index(drop=True)

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41106 entries, 0 to 41105
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   track             41106 non-null  object 
 1   artist            41106 non-null  object 
 2   uri               41106 non-null  object 
 3   danceability      41106 non-null  float64
 4   energy            41106 non-null  float64
 5   key               41106 non-null  int64  
 6   loudness          41106 non-null  float64
 7   mode              41106 non-null  int64  
 8   speechiness       41106 non-null  float64
 9   acousticness      41106 non-null  float64
 10  instrumentalness  41106 non-null  float64
 11  liveness          41106 non-null  float64
 12  valence           41106 non-null  float64
 13  tempo             41106 non-null  float64
 14  duration_ms       41106 non-null  int64  
 15  time_signature    41106 non-null  int64  
 16  chorus_hit        41106 non-null  float64
 17  sections          41106 non-null  int64  
 18  target            41106 non-null  int64  
 19  decade            41106 non-null  int64  
dtypes: float64(10), int64(7), object(3)
memory usage: 6.3+ MB
```

Remove the features "track", "artist" and "uri" as unnecessary

While intersting to know the identity of the songs labeled as a Hit or Flop, these data are not needed for training purposes.

In [5]:
```python
data = data.drop(['track', 'artist', 'uri'], axis=1)

data.describe()
```

Out[5]:

	danceability	energy	key	loudness	mode	speechiness	acou
count	41106.000000	41106.000000	41106.000000	41106.000000	41106.000000	41106.000000	41106
mean	0.539695	0.579545	5.213594	-10.221525	0.693354	0.072960	C
std	0.177821	0.252628	3.534977	5.311626	0.461107	0.086112	C
min	0.000000	0.000251	0.000000	-49.253000	0.000000	0.000000	C
25%	0.420000	0.396000	2.000000	-12.816000	0.000000	0.033700	C
50%	0.552000	0.601000	5.000000	-9.257000	1.000000	0.043400	C
75%	0.669000	0.787000	8.000000	-6.374250	1.000000	0.069800	C
max	0.988000	1.000000	11.000000	3.744000	1.000000	0.960000	C

Let's check for nulls

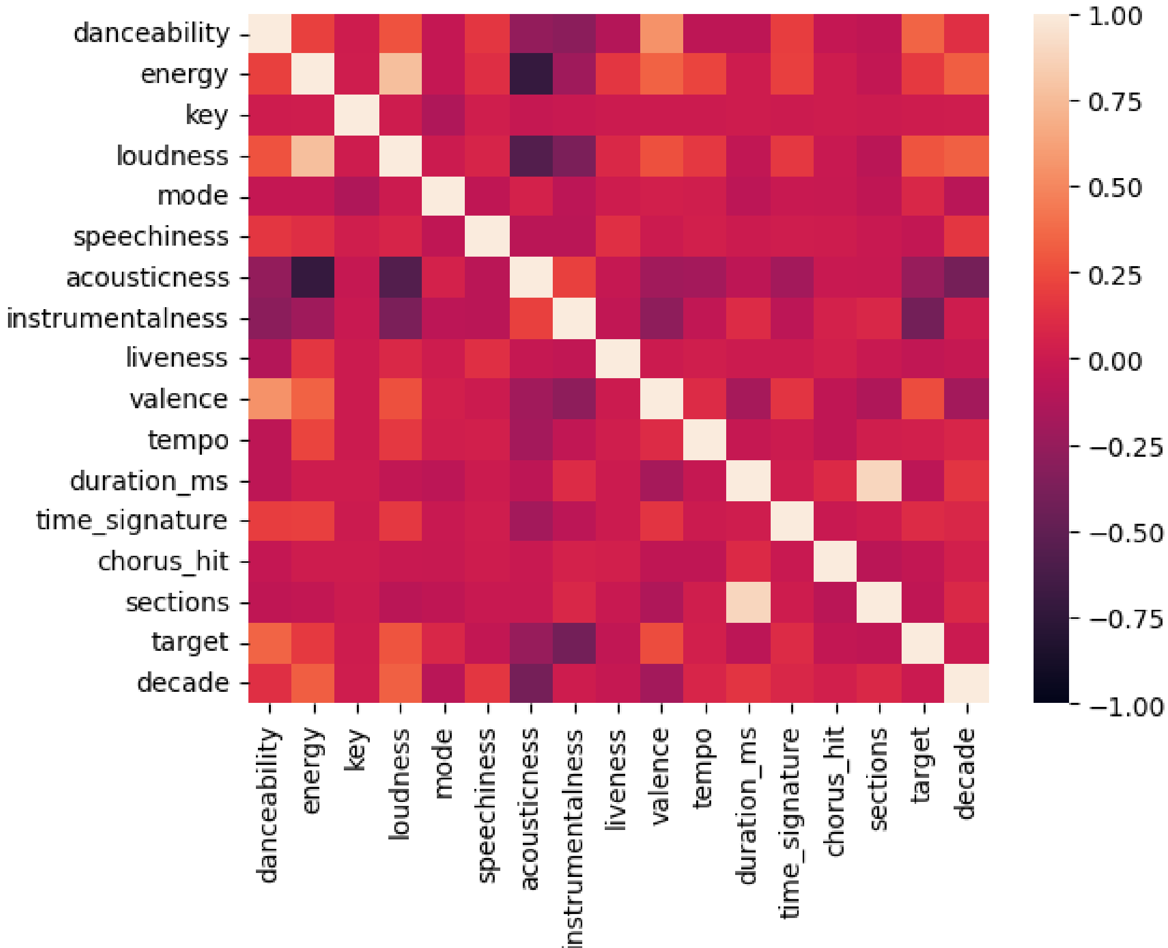
In [6]:
```
for c in data.columns:
    print(c, data[c].isnull().sum())
```

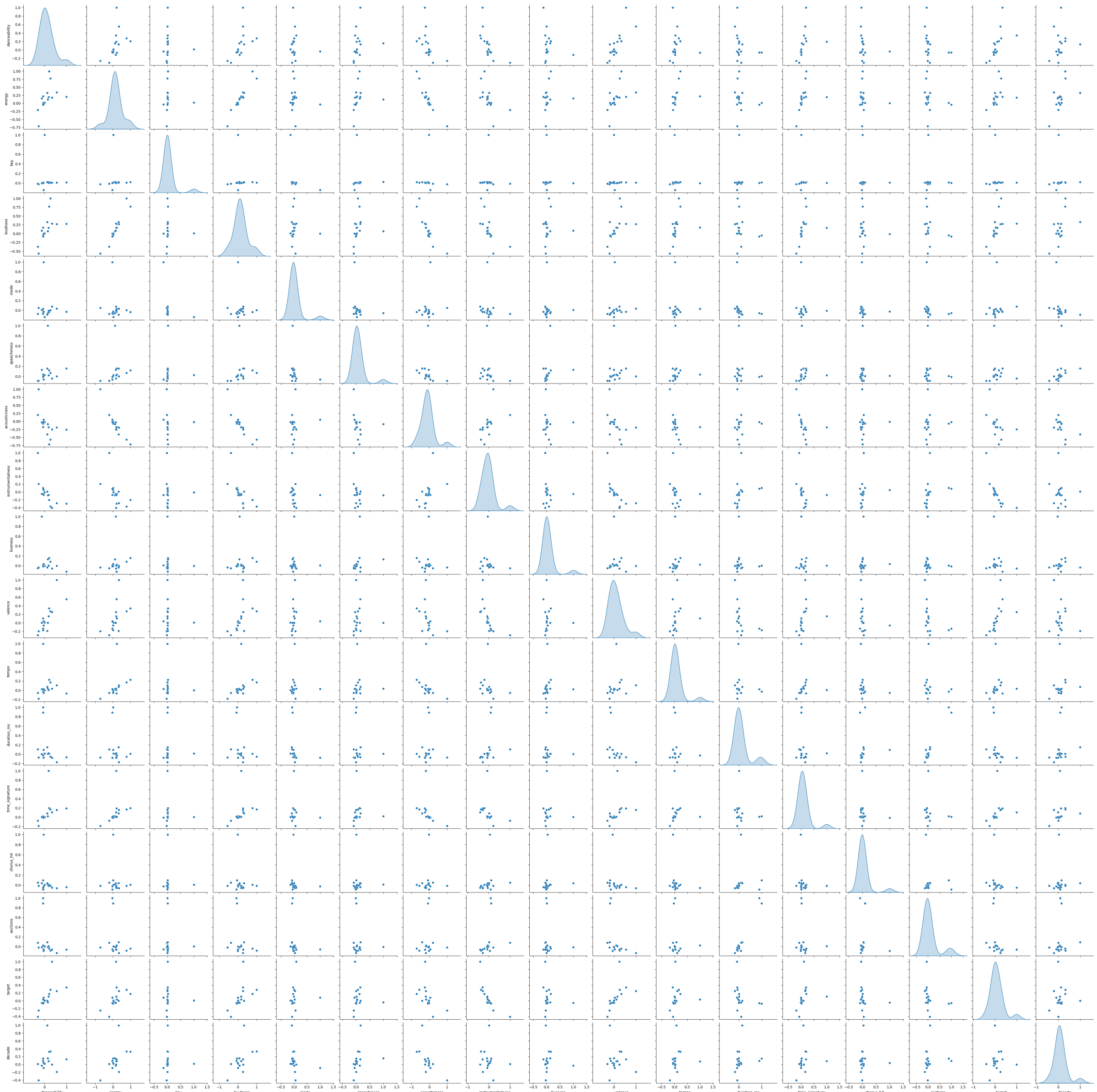
```
danceability 0
energy 0
key 0
loudness 0
mode 0
speechiness 0
acousticness 0
instrumentalness 0
liveness 0
valence 0
tempo 0
duration_ms 0
time_signature 0
chorus_hit 0
sections 0
target 0
decade 0
```

No nulls. Now we check to see if any features are already highly correlated to the target feature

In [7]:
```
import seaborn as sns

corr = data.corr()
print(corr['target'].sort_values())

heatmap = sns.heatmap(corr, vmin=-1, vmax=1, annot=False)
pairplot = sns.pairplot(corr, diag_kind='kde')
```

```
instrumentalness   -4.076382e-01
acousticness       -2.460359e-01
duration_ms        -7.381953e-02
sections           -5.999705e-02
liveness           -5.144505e-02
chorus_hit         -4.640850e-02
speechiness        -4.083547e-02
decade             -6.930299e-16
key                 9.882525e-03
tempo               3.264878e-02
mode                7.961369e-02
time_signature      1.048840e-01
energy              1.771423e-01
valence             2.511466e-01
loudness            2.860341e-01
danceability        3.460966e-01
target              1.000000e+00
Name: target, dtype: float64
```


danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
duration_ms
time_signature
chorus_hit
sections
target
decade
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



Looking at the row for "target", no other feature is either positively or negatively correlated. "danceability" and "loudness" has the highest positive correaltion of approximately 0.3, while "instrumentalness" has the highest negative correlation at about 0.4.

In the pair plot, no two features look obviously collinear. Therefore we will leave all features in for training.

Split the data into Input matrix X and Target vector Y

In [8]:

```
X = data.drop('target',axis=1)
Y = data['target']
```

Now we perform a simple Linear Regression analysis

In [9]:

```
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

```
model_multi = smf.ols(formula="target ~ "+"+".join(X.columns), data=data).fit()
model_multi.summary()
```

Out[9]:

OLS Regression Results

Dep. Variable:	target	**R-squared:**	0.267
Model:	OLS	**Adj. R-squared:**	0.267
Method:	Least Squares	**F-statistic:**	937.7
Date:	Mon, 05 Dec 2022	**Prob (F-statistic):**	0.00
Time:	20:33:09	**Log-Likelihood:**	-23437.
No. Observations:	41106	**AIC:**	4.691e+04
Df Residuals:	41089	**BIC:**	4.706e+04
Df Model:	16		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	5.9335	0.296	20.047	0.000	5.353	6.514
danceability	0.6415	0.016	39.247	0.000	0.610	0.674
energy	-0.2657	0.017	-15.345	0.000	-0.300	-0.232
key	0.0018	0.001	2.989	0.003	0.001	0.003
loudness	0.0149	0.001	21.499	0.000	0.014	0.016
mode	0.0632	0.005	13.498	0.000	0.054	0.072
speechiness	-0.5111	0.026	-19.814	0.000	-0.562	-0.461
acousticness	-0.2735	0.010	-28.666	0.000	-0.292	-0.255
instrumentalness	-0.4492	0.008	-56.027	0.000	-0.465	-0.433
liveness	-0.0827	0.013	-6.484	0.000	-0.108	-0.058
valence	-0.0270	0.011	-2.364	0.018	-0.049	-0.005
tempo	0.0002	7.66e-05	2.841	0.005	6.75e-05	0.000
duration_ms	-8.996e-08	4.3e-08	-2.092	0.036	-1.74e-07	-5.66e-09
time_signature	0.0211	0.005	4.090	0.000	0.011	0.031
chorus_hit	-0.0004	0.000	-3.089	0.002	-0.001	-0.000
sections	0.0006	0.001	0.597	0.551	-0.001	0.003
decade	-0.0027	0.000	-18.243	0.000	-0.003	-0.002

Omnibus:	165907.213	**Durbin-Watson:**	1.935
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3570.310
Skew:	-0.312	**Prob(JB):**	0.00
Kurtosis:	1.698	**Cond. No.**	3.69e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.69e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

The "sections" feature has a high p-value indicating that it is not a significant linear feature.

Further split the X and Y into train/test sets with 20% retained for testing

In [10]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test  = train_test_split(X, Y, train_size=0.8, random_s
```

Model Analysis

Let's run through a collection of standard classification models with their default parameters

In [11]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

classifiers = {
    "Logistic Regression": LogisticRegression(),
    "      Decision Tree": DecisionTreeClassifier(),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "   SVM (RBF Kernel)": SVC(),
    "     Neural Network": MLPClassifier(),
    "      Random Forest": RandomForestClassifier(),
    "     Gradient Boost": GradientBoostingClassifier()
}
for classifier_name, classifier in classifiers.items():
    classifier.fit(X_train,Y_train)
    print(classifier_name + ": training set score:  {:.2f}% - test set score: {:.2f
```

```
Logistic Regression: training set score:  49.91% - test set score: 50.64%
      Decision Tree: training set score:  99.96% - test set score: 72.44%
K-Nearest Neighbors: training set score:  72.61% - test set score: 58.45%
   SVM (RBF Kernel): training set score:  60.02% - test set score: 60.76%
     Neural Network: training set score:  50.57% - test set score: 49.72%
      Random Forest: training set score:  99.96% - test set score: 80.72%
     Gradient Boost: training set score:  80.13% - test set score: 79.59%
```

Random Forest did quite well. Let's optimize

Grid Search parameter space

In [12]:
```
from sklearn.model_selection import GridSearchCV

classifier = RandomForestClassifier()

param_grid = {
    'n_estimators': [100,200],
    'max_features': ['sqrt','log2'],
    'max_depth' : [1,2,4,8,16,32,64,None],
    'criterion' : ['gini','entropy','log_loss']
}

print('Parameter Space:',param_grid)

grid = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=2, verbose=4, s
grid.fit(X_train, Y_train)

print('Best Parameters: ', grid.best_params_)
print('Score of Best Parameters: ', grid.best_score_)
print('Score of Best Parameters on training set: ', grid.best_estimator_.score(X_tr
print('Score of Best Parameters on test set: ', grid.best_estimator_.score(X_test,
```

```
Parameter Space: {'n_estimators': [100, 200], 'max_features': ['sqrt', 'log2'], 'm
ax_depth': [1, 2, 4, 8, 16, 32, 64, None], 'criterion': ['gini', 'entropy', 'log_l
oss']}
Fitting 2 folds for each of 96 candidates, totalling 192 fits
[CV 1/2] END criterion=gini, max_depth=1, max_features=sqrt, n_estimators=100;, sc
ore=0.728 total time=   0.3s
[CV 2/2] END criterion=gini, max_depth=1, max_features=sqrt, n_estimators=100;, sc
ore=0.718 total time=   0.3s
[CV 1/2] END criterion=gini, max_depth=1, max_features=sqrt, n_estimators=200;, sc
ore=0.726 total time=   0.7s
[CV 2/2] END criterion=gini, max_depth=1, max_features=sqrt, n_estimators=200;, sc
ore=0.720 total time=   0.8s
[CV 1/2] END criterion=gini, max_depth=1, max_features=log2, n_estimators=100;, sc
ore=0.723 total time=   0.3s
[CV 2/2] END criterion=gini, max_depth=1, max_features=log2, n_estimators=100;, sc
ore=0.714 total time=   0.3s
[CV 1/2] END criterion=gini, max_depth=1, max_features=log2, n_estimators=200;, sc
ore=0.721 total time=   0.7s
[CV 2/2] END criterion=gini, max_depth=1, max_features=log2, n_estimators=200;, sc
ore=0.717 total time=   0.7s
[CV 1/2] END criterion=gini, max_depth=2, max_features=sqrt, n_estimators=100;, sc
ore=0.747 total time=   0.5s
[CV 2/2] END criterion=gini, max_depth=2, max_features=sqrt, n_estimators=100;, sc
ore=0.730 total time=   0.5s
[CV 1/2] END criterion=gini, max_depth=2, max_features=sqrt, n_estimators=200;, sc
ore=0.743 total time=   1.2s
[CV 2/2] END criterion=gini, max_depth=2, max_features=sqrt, n_estimators=200;, sc
ore=0.736 total time=   1.1s
[CV 1/2] END criterion=gini, max_depth=2, max_features=log2, n_estimators=100;, sc
ore=0.745 total time=   0.5s
[CV 2/2] END criterion=gini, max_depth=2, max_features=log2, n_estimators=100;, sc
ore=0.727 total time=   0.5s
[CV 1/2] END criterion=gini, max_depth=2, max_features=log2, n_estimators=200;, sc
ore=0.744 total time=   1.1s
[CV 2/2] END criterion=gini, max_depth=2, max_features=log2, n_estimators=200;, sc
ore=0.735 total time=   1.2s
[CV 1/2] END criterion=gini, max_depth=4, max_features=sqrt, n_estimators=100;, sc
ore=0.759 total time=   1.0s
[CV 2/2] END criterion=gini, max_depth=4, max_features=sqrt, n_estimators=100;, sc
ore=0.756 total time=   0.9s
[CV 1/2] END criterion=gini, max_depth=4, max_features=sqrt, n_estimators=200;, sc
ore=0.758 total time=   2.0s
[CV 2/2] END criterion=gini, max_depth=4, max_features=sqrt, n_estimators=200;, sc
ore=0.755 total time=   2.0s
[CV 1/2] END criterion=gini, max_depth=4, max_features=log2, n_estimators=100;, sc
ore=0.759 total time=   1.0s
[CV 2/2] END criterion=gini, max_depth=4, max_features=log2, n_estimators=100;, sc
ore=0.756 total time=   1.0s
[CV 1/2] END criterion=gini, max_depth=4, max_features=log2, n_estimators=200;, sc
ore=0.760 total time=   2.0s
[CV 2/2] END criterion=gini, max_depth=4, max_features=log2, n_estimators=200;, sc
ore=0.756 total time=   2.0s
[CV 1/2] END criterion=gini, max_depth=8, max_features=sqrt, n_estimators=100;, sc
ore=0.778 total time=   1.8s
[CV 2/2] END criterion=gini, max_depth=8, max_features=sqrt, n_estimators=100;, sc
ore=0.776 total time=   1.8s
```

```
[CV 1/2] END criterion=gini, max_depth=8, max_features=sqrt, n_estimators=200;, sc
ore=0.778 total time=   3.7s
[CV 2/2] END criterion=gini, max_depth=8, max_features=sqrt, n_estimators=200;, sc
ore=0.774 total time=   3.7s
[CV 1/2] END criterion=gini, max_depth=8, max_features=log2, n_estimators=100;, sc
ore=0.780 total time=   1.8s
[CV 2/2] END criterion=gini, max_depth=8, max_features=log2, n_estimators=100;, sc
ore=0.775 total time=   1.8s
[CV 1/2] END criterion=gini, max_depth=8, max_features=log2, n_estimators=200;, sc
ore=0.777 total time=   3.6s
[CV 2/2] END criterion=gini, max_depth=8, max_features=log2, n_estimators=200;, sc
ore=0.776 total time=   3.6s
[CV 1/2] END criterion=gini, max_depth=16, max_features=sqrt, n_estimators=100;, s
core=0.797 total time=   2.9s
[CV 2/2] END criterion=gini, max_depth=16, max_features=sqrt, n_estimators=100;, s
core=0.793 total time=   2.9s
[CV 1/2] END criterion=gini, max_depth=16, max_features=sqrt, n_estimators=200;, s
core=0.801 total time=   6.0s
[CV 2/2] END criterion=gini, max_depth=16, max_features=sqrt, n_estimators=200;, s
core=0.792 total time=   6.0s
[CV 1/2] END criterion=gini, max_depth=16, max_features=log2, n_estimators=100;, s
core=0.799 total time=   2.9s
[CV 2/2] END criterion=gini, max_depth=16, max_features=log2, n_estimators=100;, s
core=0.791 total time=   2.9s
[CV 1/2] END criterion=gini, max_depth=16, max_features=log2, n_estimators=200;, s
core=0.799 total time=   6.0s
[CV 2/2] END criterion=gini, max_depth=16, max_features=log2, n_estimators=200;, s
core=0.795 total time=   6.0s
[CV 1/2] END criterion=gini, max_depth=32, max_features=sqrt, n_estimators=100;, s
core=0.800 total time=   3.4s
[CV 2/2] END criterion=gini, max_depth=32, max_features=sqrt, n_estimators=100;, s
core=0.793 total time=   3.4s
[CV 1/2] END criterion=gini, max_depth=32, max_features=sqrt, n_estimators=200;, s
core=0.802 total time=   6.9s
[CV 2/2] END criterion=gini, max_depth=32, max_features=sqrt, n_estimators=200;, s
core=0.795 total time=   6.9s
[CV 1/2] END criterion=gini, max_depth=32, max_features=log2, n_estimators=100;, s
core=0.801 total time=   3.4s
[CV 2/2] END criterion=gini, max_depth=32, max_features=log2, n_estimators=100;, s
core=0.793 total time=   3.4s
[CV 1/2] END criterion=gini, max_depth=32, max_features=log2, n_estimators=200;, s
core=0.801 total time=   6.8s
[CV 2/2] END criterion=gini, max_depth=32, max_features=log2, n_estimators=200;, s
core=0.794 total time=   6.9s
[CV 1/2] END criterion=gini, max_depth=64, max_features=sqrt, n_estimators=100;, s
core=0.799 total time=   3.4s
[CV 2/2] END criterion=gini, max_depth=64, max_features=sqrt, n_estimators=100;, s
core=0.795 total time=   3.4s
[CV 1/2] END criterion=gini, max_depth=64, max_features=sqrt, n_estimators=200;, s
core=0.799 total time=   6.9s
[CV 2/2] END criterion=gini, max_depth=64, max_features=sqrt, n_estimators=200;, s
core=0.794 total time=   6.9s
[CV 1/2] END criterion=gini, max_depth=64, max_features=log2, n_estimators=100;, s
core=0.798 total time=   3.4s
[CV 2/2] END criterion=gini, max_depth=64, max_features=log2, n_estimators=100;, s
core=0.794 total time=   3.4s
```

```
[CV 1/2] END criterion=gini, max_depth=64, max_features=log2, n_estimators=200;, s
core=0.801 total time=   7.1s
[CV 2/2] END criterion=gini, max_depth=64, max_features=log2, n_estimators=200;, s
core=0.795 total time=   7.0s
[CV 1/2] END criterion=gini, max_depth=None, max_features=sqrt, n_estimators=100;,
score=0.799 total time=   3.4s
[CV 2/2] END criterion=gini, max_depth=None, max_features=sqrt, n_estimators=100;,
score=0.796 total time=   3.5s
[CV 1/2] END criterion=gini, max_depth=None, max_features=sqrt, n_estimators=200;,
score=0.803 total time=   6.9s
[CV 2/2] END criterion=gini, max_depth=None, max_features=sqrt, n_estimators=200;,
score=0.796 total time=   7.2s
[CV 1/2] END criterion=gini, max_depth=None, max_features=log2, n_estimators=100;,
score=0.799 total time=   3.5s
[CV 2/2] END criterion=gini, max_depth=None, max_features=log2, n_estimators=100;,
score=0.793 total time=   3.5s
[CV 1/2] END criterion=gini, max_depth=None, max_features=log2, n_estimators=200;,
score=0.801 total time=   6.9s
[CV 2/2] END criterion=gini, max_depth=None, max_features=log2, n_estimators=200;,
score=0.793 total time=   7.0s
[CV 1/2] END criterion=entropy, max_depth=1, max_features=sqrt, n_estimators=100;,
score=0.727 total time=   0.4s
[CV 2/2] END criterion=entropy, max_depth=1, max_features=sqrt, n_estimators=100;,
score=0.716 total time=   0.4s
[CV 1/2] END criterion=entropy, max_depth=1, max_features=sqrt, n_estimators=200;,
score=0.728 total time=   0.8s
[CV 2/2] END criterion=entropy, max_depth=1, max_features=sqrt, n_estimators=200;,
score=0.716 total time=   0.9s
[CV 1/2] END criterion=entropy, max_depth=1, max_features=log2, n_estimators=100;,
score=0.722 total time=   0.4s
[CV 2/2] END criterion=entropy, max_depth=1, max_features=log2, n_estimators=100;,
score=0.726 total time=   0.4s
[CV 1/2] END criterion=entropy, max_depth=1, max_features=log2, n_estimators=200;,
score=0.723 total time=   0.9s
[CV 2/2] END criterion=entropy, max_depth=1, max_features=log2, n_estimators=200;,
score=0.715 total time=   0.9s
[CV 1/2] END criterion=entropy, max_depth=2, max_features=sqrt, n_estimators=100;,
score=0.741 total time=   0.7s
[CV 2/2] END criterion=entropy, max_depth=2, max_features=sqrt, n_estimators=100;,
score=0.733 total time=   0.7s
[CV 1/2] END criterion=entropy, max_depth=2, max_features=sqrt, n_estimators=200;,
score=0.740 total time=   1.4s
[CV 2/2] END criterion=entropy, max_depth=2, max_features=sqrt, n_estimators=200;,
score=0.731 total time=   1.4s
[CV 1/2] END criterion=entropy, max_depth=2, max_features=log2, n_estimators=100;,
score=0.747 total time=   0.7s
[CV 2/2] END criterion=entropy, max_depth=2, max_features=log2, n_estimators=100;,
score=0.727 total time=   0.7s
[CV 1/2] END criterion=entropy, max_depth=2, max_features=log2, n_estimators=200;,
score=0.742 total time=   1.4s
[CV 2/2] END criterion=entropy, max_depth=2, max_features=log2, n_estimators=200;,
score=0.734 total time=   1.4s
[CV 1/2] END criterion=entropy, max_depth=4, max_features=sqrt, n_estimators=100;,
score=0.757 total time=   1.2s
[CV 2/2] END criterion=entropy, max_depth=4, max_features=sqrt, n_estimators=100;,
score=0.753 total time=   1.3s
```

```
[CV 1/2] END criterion=entropy, max_depth=4, max_features=sqrt, n_estimators=200;,
score=0.758 total time=   2.5s
[CV 2/2] END criterion=entropy, max_depth=4, max_features=sqrt, n_estimators=200;,
score=0.755 total time=   2.5s
[CV 1/2] END criterion=entropy, max_depth=4, max_features=log2, n_estimators=100;,
score=0.758 total time=   1.2s
[CV 2/2] END criterion=entropy, max_depth=4, max_features=log2, n_estimators=100;,
score=0.753 total time=   1.2s
[CV 1/2] END criterion=entropy, max_depth=4, max_features=log2, n_estimators=200;,
score=0.757 total time=   2.5s
[CV 2/2] END criterion=entropy, max_depth=4, max_features=log2, n_estimators=200;,
score=0.755 total time=   2.6s
[CV 1/2] END criterion=entropy, max_depth=8, max_features=sqrt, n_estimators=100;,
score=0.776 total time=   2.4s
[CV 2/2] END criterion=entropy, max_depth=8, max_features=sqrt, n_estimators=100;,
score=0.775 total time=   2.4s
[CV 1/2] END criterion=entropy, max_depth=8, max_features=sqrt, n_estimators=200;,
score=0.778 total time=   4.8s
[CV 2/2] END criterion=entropy, max_depth=8, max_features=sqrt, n_estimators=200;,
score=0.774 total time=   4.8s
[CV 1/2] END criterion=entropy, max_depth=8, max_features=log2, n_estimators=100;,
score=0.775 total time=   2.4s
[CV 2/2] END criterion=entropy, max_depth=8, max_features=log2, n_estimators=100;,
score=0.775 total time=   2.4s
[CV 1/2] END criterion=entropy, max_depth=8, max_features=log2, n_estimators=200;,
score=0.777 total time=   4.8s
[CV 2/2] END criterion=entropy, max_depth=8, max_features=log2, n_estimators=200;,
score=0.776 total time=   4.8s
[CV 1/2] END criterion=entropy, max_depth=16, max_features=sqrt, n_estimators=10
0;, score=0.800 total time=   4.2s
[CV 2/2] END criterion=entropy, max_depth=16, max_features=sqrt, n_estimators=10
0;, score=0.791 total time=   4.1s
[CV 1/2] END criterion=entropy, max_depth=16, max_features=sqrt, n_estimators=20
0;, score=0.801 total time=   8.4s
[CV 2/2] END criterion=entropy, max_depth=16, max_features=sqrt, n_estimators=20
0;, score=0.794 total time=   8.3s
[CV 1/2] END criterion=entropy, max_depth=16, max_features=log2, n_estimators=10
0;, score=0.797 total time=   4.1s
[CV 2/2] END criterion=entropy, max_depth=16, max_features=log2, n_estimators=10
0;, score=0.792 total time=   4.1s
[CV 1/2] END criterion=entropy, max_depth=16, max_features=log2, n_estimators=20
0;, score=0.800 total time=   8.4s
[CV 2/2] END criterion=entropy, max_depth=16, max_features=log2, n_estimators=20
0;, score=0.792 total time=   8.4s
[CV 1/2] END criterion=entropy, max_depth=32, max_features=sqrt, n_estimators=10
0;, score=0.799 total time=   4.7s
[CV 2/2] END criterion=entropy, max_depth=32, max_features=sqrt, n_estimators=10
0;, score=0.795 total time=   4.8s
[CV 1/2] END criterion=entropy, max_depth=32, max_features=sqrt, n_estimators=20
0;, score=0.802 total time=   9.7s
[CV 2/2] END criterion=entropy, max_depth=32, max_features=sqrt, n_estimators=20
0;, score=0.795 total time=   9.8s
[CV 1/2] END criterion=entropy, max_depth=32, max_features=log2, n_estimators=10
0;, score=0.798 total time=   4.7s
[CV 2/2] END criterion=entropy, max_depth=32, max_features=log2, n_estimators=10
0;, score=0.794 total time=   4.8s
```

```
[CV 1/2] END criterion=entropy, max_depth=32, max_features=log2, n_estimators=20
0;, score=0.802 total time=   9.6s
[CV 2/2] END criterion=entropy, max_depth=32, max_features=log2, n_estimators=20
0;, score=0.796 total time=   9.6s
[CV 1/2] END criterion=entropy, max_depth=64, max_features=sqrt, n_estimators=10
0;, score=0.801 total time=   4.8s
[CV 2/2] END criterion=entropy, max_depth=64, max_features=sqrt, n_estimators=10
0;, score=0.796 total time=   4.8s
[CV 1/2] END criterion=entropy, max_depth=64, max_features=sqrt, n_estimators=20
0;, score=0.801 total time=   9.6s
[CV 2/2] END criterion=entropy, max_depth=64, max_features=sqrt, n_estimators=20
0;, score=0.794 total time=   9.9s
[CV 1/2] END criterion=entropy, max_depth=64, max_features=log2, n_estimators=10
0;, score=0.800 total time=   4.8s
[CV 2/2] END criterion=entropy, max_depth=64, max_features=log2, n_estimators=10
0;, score=0.796 total time=   4.7s
[CV 1/2] END criterion=entropy, max_depth=64, max_features=log2, n_estimators=20
0;, score=0.803 total time=   9.6s
[CV 2/2] END criterion=entropy, max_depth=64, max_features=log2, n_estimators=20
0;, score=0.796 total time=   9.6s
[CV 1/2] END criterion=entropy, max_depth=None, max_features=sqrt, n_estimators=10
0;, score=0.798 total time=   4.8s
[CV 2/2] END criterion=entropy, max_depth=None, max_features=sqrt, n_estimators=10
0;, score=0.794 total time=   4.8s
[CV 1/2] END criterion=entropy, max_depth=None, max_features=sqrt, n_estimators=20
0;, score=0.802 total time=   9.6s
[CV 2/2] END criterion=entropy, max_depth=None, max_features=sqrt, n_estimators=20
0;, score=0.795 total time=   9.7s
[CV 1/2] END criterion=entropy, max_depth=None, max_features=log2, n_estimators=10
0;, score=0.800 total time=   4.8s
[CV 2/2] END criterion=entropy, max_depth=None, max_features=log2, n_estimators=10
0;, score=0.794 total time=   4.8s
[CV 1/2] END criterion=entropy, max_depth=None, max_features=log2, n_estimators=20
0;, score=0.801 total time=   9.6s
[CV 2/2] END criterion=entropy, max_depth=None, max_features=log2, n_estimators=20
0;, score=0.797 total time=   9.7s
[CV 1/2] END criterion=log_loss, max_depth=1, max_features=sqrt, n_estimators=10
0;, score=0.725 total time=   0.4s
[CV 2/2] END criterion=log_loss, max_depth=1, max_features=sqrt, n_estimators=10
0;, score=0.718 total time=   0.4s
[CV 1/2] END criterion=log_loss, max_depth=1, max_features=sqrt, n_estimators=20
0;, score=0.715 total time=   0.9s
[CV 2/2] END criterion=log_loss, max_depth=1, max_features=sqrt, n_estimators=20
0;, score=0.709 total time=   0.9s
[CV 1/2] END criterion=log_loss, max_depth=1, max_features=log2, n_estimators=10
0;, score=0.724 total time=   0.4s
[CV 2/2] END criterion=log_loss, max_depth=1, max_features=log2, n_estimators=10
0;, score=0.716 total time=   0.4s
[CV 1/2] END criterion=log_loss, max_depth=1, max_features=log2, n_estimators=20
0;, score=0.725 total time=   0.9s
[CV 2/2] END criterion=log_loss, max_depth=1, max_features=log2, n_estimators=20
0;, score=0.720 total time=   0.9s
[CV 1/2] END criterion=log_loss, max_depth=2, max_features=sqrt, n_estimators=10
0;, score=0.745 total time=   0.7s
[CV 2/2] END criterion=log_loss, max_depth=2, max_features=sqrt, n_estimators=10
0;, score=0.733 total time=   0.7s
```

```
[CV 1/2] END criterion=log_loss, max_depth=2, max_features=sqrt, n_estimators=20
0;, score=0.746 total time=   1.4s
[CV 2/2] END criterion=log_loss, max_depth=2, max_features=sqrt, n_estimators=20
0;, score=0.723 total time=   1.4s
[CV 1/2] END criterion=log_loss, max_depth=2, max_features=log2, n_estimators=10
0;, score=0.741 total time=   0.7s
[CV 2/2] END criterion=log_loss, max_depth=2, max_features=log2, n_estimators=10
0;, score=0.733 total time=   0.7s
[CV 1/2] END criterion=log_loss, max_depth=2, max_features=log2, n_estimators=20
0;, score=0.737 total time=   1.4s
[CV 2/2] END criterion=log_loss, max_depth=2, max_features=log2, n_estimators=20
0;, score=0.727 total time=   1.4s
[CV 1/2] END criterion=log_loss, max_depth=4, max_features=sqrt, n_estimators=10
0;, score=0.756 total time=   1.2s
[CV 2/2] END criterion=log_loss, max_depth=4, max_features=sqrt, n_estimators=10
0;, score=0.755 total time=   1.2s
[CV 1/2] END criterion=log_loss, max_depth=4, max_features=sqrt, n_estimators=20
0;, score=0.756 total time=   2.6s
[CV 2/2] END criterion=log_loss, max_depth=4, max_features=sqrt, n_estimators=20
0;, score=0.756 total time=   2.5s
[CV 1/2] END criterion=log_loss, max_depth=4, max_features=log2, n_estimators=10
0;, score=0.759 total time=   1.2s
[CV 2/2] END criterion=log_loss, max_depth=4, max_features=log2, n_estimators=10
0;, score=0.755 total time=   1.2s
[CV 1/2] END criterion=log_loss, max_depth=4, max_features=log2, n_estimators=20
0;, score=0.756 total time=   2.6s
[CV 2/2] END criterion=log_loss, max_depth=4, max_features=log2, n_estimators=20
0;, score=0.753 total time=   2.6s
[CV 1/2] END criterion=log_loss, max_depth=8, max_features=sqrt, n_estimators=10
0;, score=0.776 total time=   2.4s
[CV 2/2] END criterion=log_loss, max_depth=8, max_features=sqrt, n_estimators=10
0;, score=0.773 total time=   2.4s
[CV 1/2] END criterion=log_loss, max_depth=8, max_features=sqrt, n_estimators=20
0;, score=0.776 total time=   4.8s
[CV 2/2] END criterion=log_loss, max_depth=8, max_features=sqrt, n_estimators=20
0;, score=0.774 total time=   4.8s
[CV 1/2] END criterion=log_loss, max_depth=8, max_features=log2, n_estimators=10
0;, score=0.776 total time=   2.4s
[CV 2/2] END criterion=log_loss, max_depth=8, max_features=log2, n_estimators=10
0;, score=0.775 total time=   2.4s
[CV 1/2] END criterion=log_loss, max_depth=8, max_features=log2, n_estimators=20
0;, score=0.776 total time=   4.9s
[CV 2/2] END criterion=log_loss, max_depth=8, max_features=log2, n_estimators=20
0;, score=0.774 total time=   4.8s
[CV 1/2] END criterion=log_loss, max_depth=16, max_features=sqrt, n_estimators=10
0;, score=0.799 total time=   4.1s
[CV 2/2] END criterion=log_loss, max_depth=16, max_features=sqrt, n_estimators=10
0;, score=0.791 total time=   4.1s
[CV 1/2] END criterion=log_loss, max_depth=16, max_features=sqrt, n_estimators=20
0;, score=0.801 total time=   8.3s
[CV 2/2] END criterion=log_loss, max_depth=16, max_features=sqrt, n_estimators=20
0;, score=0.794 total time=   8.3s
[CV 1/2] END criterion=log_loss, max_depth=16, max_features=log2, n_estimators=10
0;, score=0.800 total time=   4.1s
[CV 2/2] END criterion=log_loss, max_depth=16, max_features=log2, n_estimators=10
0;, score=0.793 total time=   4.1s
```

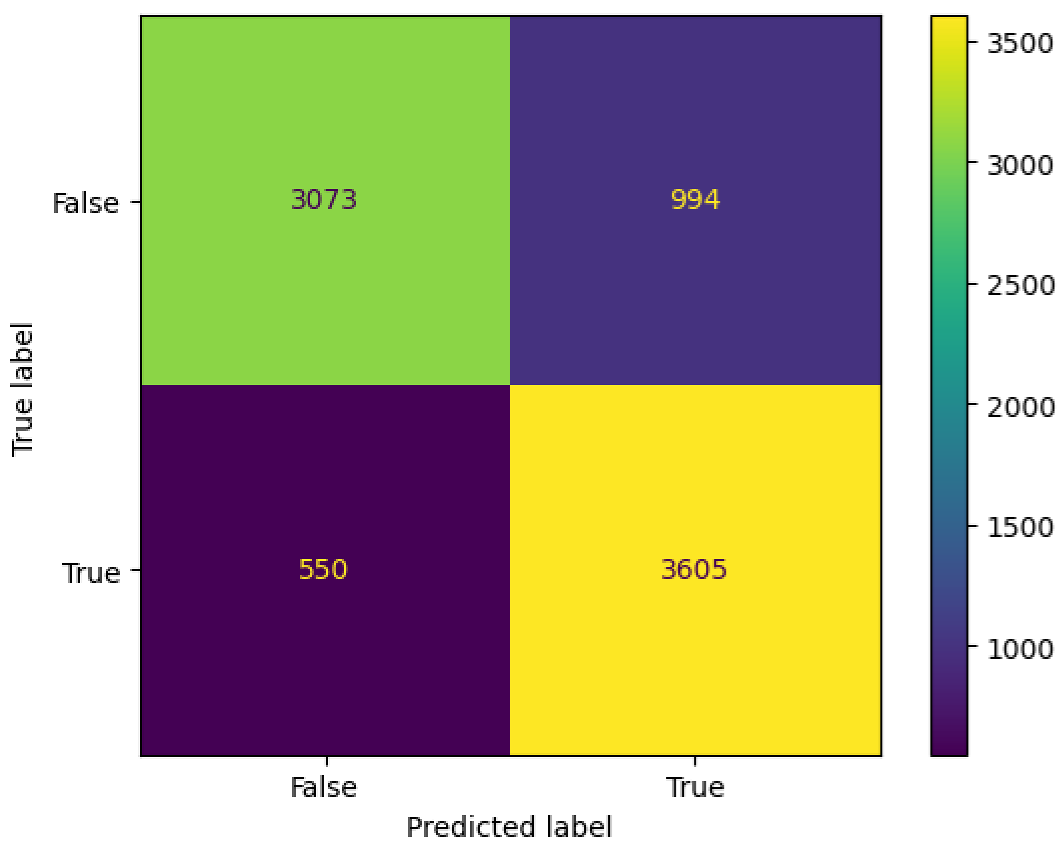
```
[CV 1/2] END criterion=log_loss, max_depth=16, max_features=log2, n_estimators=20
0;, score=0.799 total time=   8.4s
[CV 2/2] END criterion=log_loss, max_depth=16, max_features=log2, n_estimators=20
0;, score=0.793 total time=   8.4s
[CV 1/2] END criterion=log_loss, max_depth=32, max_features=sqrt, n_estimators=10
0;, score=0.802 total time=   4.7s
[CV 2/2] END criterion=log_loss, max_depth=32, max_features=sqrt, n_estimators=10
0;, score=0.796 total time=   4.8s
[CV 1/2] END criterion=log_loss, max_depth=32, max_features=sqrt, n_estimators=20
0;, score=0.803 total time=   9.6s
[CV 2/2] END criterion=log_loss, max_depth=32, max_features=sqrt, n_estimators=20
0;, score=0.796 total time=   9.7s
[CV 1/2] END criterion=log_loss, max_depth=32, max_features=log2, n_estimators=10
0;, score=0.803 total time=   5.1s
[CV 2/2] END criterion=log_loss, max_depth=32, max_features=log2, n_estimators=10
0;, score=0.796 total time=   4.8s
[CV 1/2] END criterion=log_loss, max_depth=32, max_features=log2, n_estimators=20
0;, score=0.800 total time=   9.8s
[CV 2/2] END criterion=log_loss, max_depth=32, max_features=log2, n_estimators=20
0;, score=0.794 total time=   9.8s
[CV 1/2] END criterion=log_loss, max_depth=64, max_features=sqrt, n_estimators=10
0;, score=0.802 total time=   4.8s
[CV 2/2] END criterion=log_loss, max_depth=64, max_features=sqrt, n_estimators=10
0;, score=0.796 total time=   4.8s
[CV 1/2] END criterion=log_loss, max_depth=64, max_features=sqrt, n_estimators=20
0;, score=0.803 total time=   9.6s
[CV 2/2] END criterion=log_loss, max_depth=64, max_features=sqrt, n_estimators=20
0;, score=0.797 total time=   9.7s
[CV 1/2] END criterion=log_loss, max_depth=64, max_features=log2, n_estimators=10
0;, score=0.800 total time=   4.7s
[CV 2/2] END criterion=log_loss, max_depth=64, max_features=log2, n_estimators=10
0;, score=0.795 total time=   4.7s
[CV 1/2] END criterion=log_loss, max_depth=64, max_features=log2, n_estimators=20
0;, score=0.801 total time=   9.6s
[CV 2/2] END criterion=log_loss, max_depth=64, max_features=log2, n_estimators=20
0;, score=0.796 total time=   9.6s
[CV 1/2] END criterion=log_loss, max_depth=None, max_features=sqrt, n_estimators=1
00;, score=0.801 total time=   4.7s
[CV 2/2] END criterion=log_loss, max_depth=None, max_features=sqrt, n_estimators=1
00;, score=0.794 total time=   4.7s
[CV 1/2] END criterion=log_loss, max_depth=None, max_features=sqrt, n_estimators=2
00;, score=0.802 total time=   9.7s
[CV 2/2] END criterion=log_loss, max_depth=None, max_features=sqrt, n_estimators=2
00;, score=0.796 total time=   9.7s
[CV 1/2] END criterion=log_loss, max_depth=None, max_features=log2, n_estimators=1
00;, score=0.799 total time=   4.7s
[CV 2/2] END criterion=log_loss, max_depth=None, max_features=log2, n_estimators=1
00;, score=0.795 total time=   4.7s
[CV 1/2] END criterion=log_loss, max_depth=None, max_features=log2, n_estimators=2
00;, score=0.802 total time=   9.6s
[CV 2/2] END criterion=log_loss, max_depth=None, max_features=log2, n_estimators=2
00;, score=0.796 total time=   9.6s
Best Parameters:  {'criterion': 'log_loss', 'max_depth': 64, 'max_features': 'sqr
t', 'n_estimators': 200}
Score of Best Parameters:  0.8002371974212383
```

```
Score of Best Parameters on training set:  0.9996350808904026
Score of Best Parameters on test set:  0.8122111408416444
```

Searching the parameter set didn't do much better than the Random Forest classifier's default parameters. So we will move forward and evalute the metrics of the best solution model.

In [13]:
```
from sklearn import metrics
import matplotlib.pyplot as plt

Y_pred = grid.best_estimator_.predict(X_test)

tn, fp, fn, tp   = metrics.confusion_matrix(Y_test, Y_pred).ravel()
confusion_matrix = metrics.confusion_matrix(Y_test, Y_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, di

print( "Test Set Accuracy (TP + TN) / (P + N) = {:.2f}%".format( 100 * (tp+tn)/len(
print( "Test Set Precision (TP) / (TP + FN) = {:.2f}%".format( 100 * (tp)/(tp+fp) )

cm_display.plot()
plt.show()
```

```
Test Set Accuracy (TP + TN) / (P + N) = 81.22%
Test Set Precision (TP) / (TP + FN) = 78.39%
```



In [ ]: