In [30]:

```
# Beginning of question #1 (Setup and Data Fetching)
import pandas as pd
import os

abs_path = os.path.join(os.path.dirname("bank-additional-full.csv"), "bank-additional-full.csv")
df = pd.read_csv(abs_path, sep = ";")
df
```

Out[30]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	n
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	n
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	n
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	n
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	n
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	n
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	n
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	n
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	n
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	

41188 rows × 21 columns

In [31]:

```
df = df.dropna()
df
```

Out[31]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1

41188 rows × 21 columns

In [32]:

```
df.dtypes
```

Out[32]:

```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

In [87]:

```python
# Beginning of question #2 (Data Preprocessing)
int_data, obj_data, float_data = [],[],[]
columns, dtypes = zip(*df.dtypes.items())
columns = list(columns)
dtypes = list(dtypes)
for i in range(len(columns)):
    if dtypes[i] == 'int64':
        int_data.append(columns[i])
    elif dtypes[i] == 'object':
        obj_data.append(columns[i])
    elif dtypes[i] == 'float64':
        float_data.append(columns[i])

print(int_data)
print(obj_data)
print(float_data)
```

```
['age', 'duration', 'campaign', 'pdays', 'previous']
['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome', 'y']
['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']
```

In [88]:

```
df = df[~df.job.str.contains("unknown")]
df = df[~df.marital.str.contains("unknown")]
df = df[~df.education.str.contains("unknown")]
df = df[~df.default.str.contains("unknown")]
df = df[~df.housing.str.contains("unknown")]
df = df[~df.loan.str.contains("unknown")]
df = df[~df.contact.str.contains("unknown")]
df = df[~df.month.str.contains("unknown")]
df = df[~df.day_of_week.str.contains("unknown")]
df = df[~df.poutcome.str.contains("unknown")]
df
```

Out[88]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	po
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	no
6	59	admin.	married	professional.course	no	no	no	telephone	may	mon	...	1	999	0	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	no
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	no
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	no
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	no
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	

30488 rows × 21 columns

In [89]:

```
df.describe()
```

Out[89]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m
count	30488.000000	30488.000000	30488.000000	30488.000000	30488.000000	30488.000000	30488.000000	30488.000000	30488.000000
mean	39.030012	259.484092	2.521451	956.332295	0.194273	-0.071510	93.523311	-40.602263	3.459938
std	10.333529	261.714262	2.720150	201.373292	0.522788	1.610399	0.585374	4.789249	1.777231
min	17.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000
25%	31.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.313000
50%	37.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.444000	-41.800000	4.856000
75%	45.000000	321.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000
max	95.000000	4918.000000	43.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000

In [90]:

```
df['y'].value_counts()
```

Out[90]:

```
no     26629
yes     3859
Name: y, dtype: int64
```

There are too many "no" values compared to "yes" in the y data column supporting an imbalanced #s of positives and negatives.

In [91]:

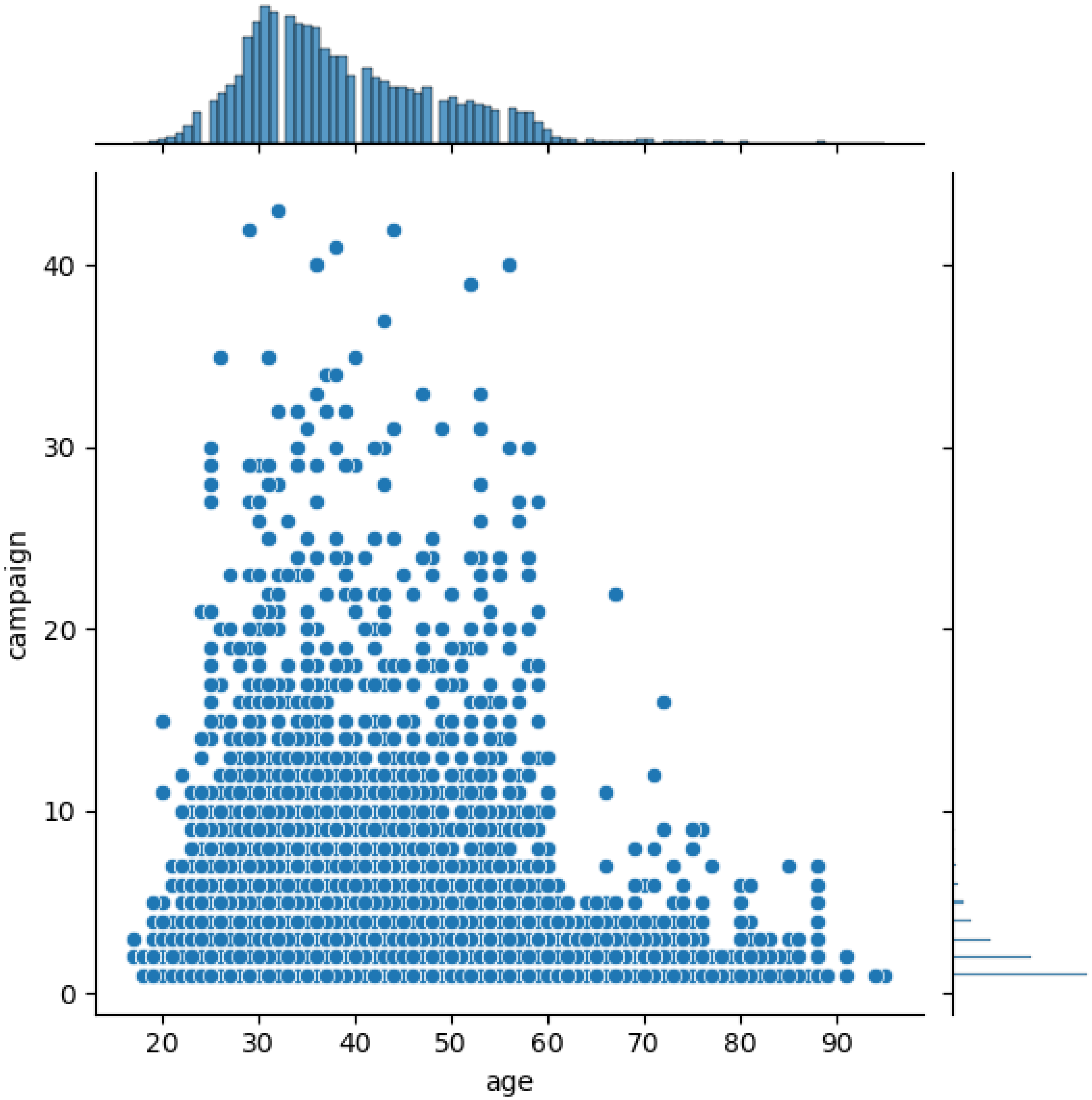
```
import seaborn as sns

age = df['age']
campaign = df['campaign']

sns.jointplot(x=age, y=campaign, kind='scatter')
```

Out[91]:

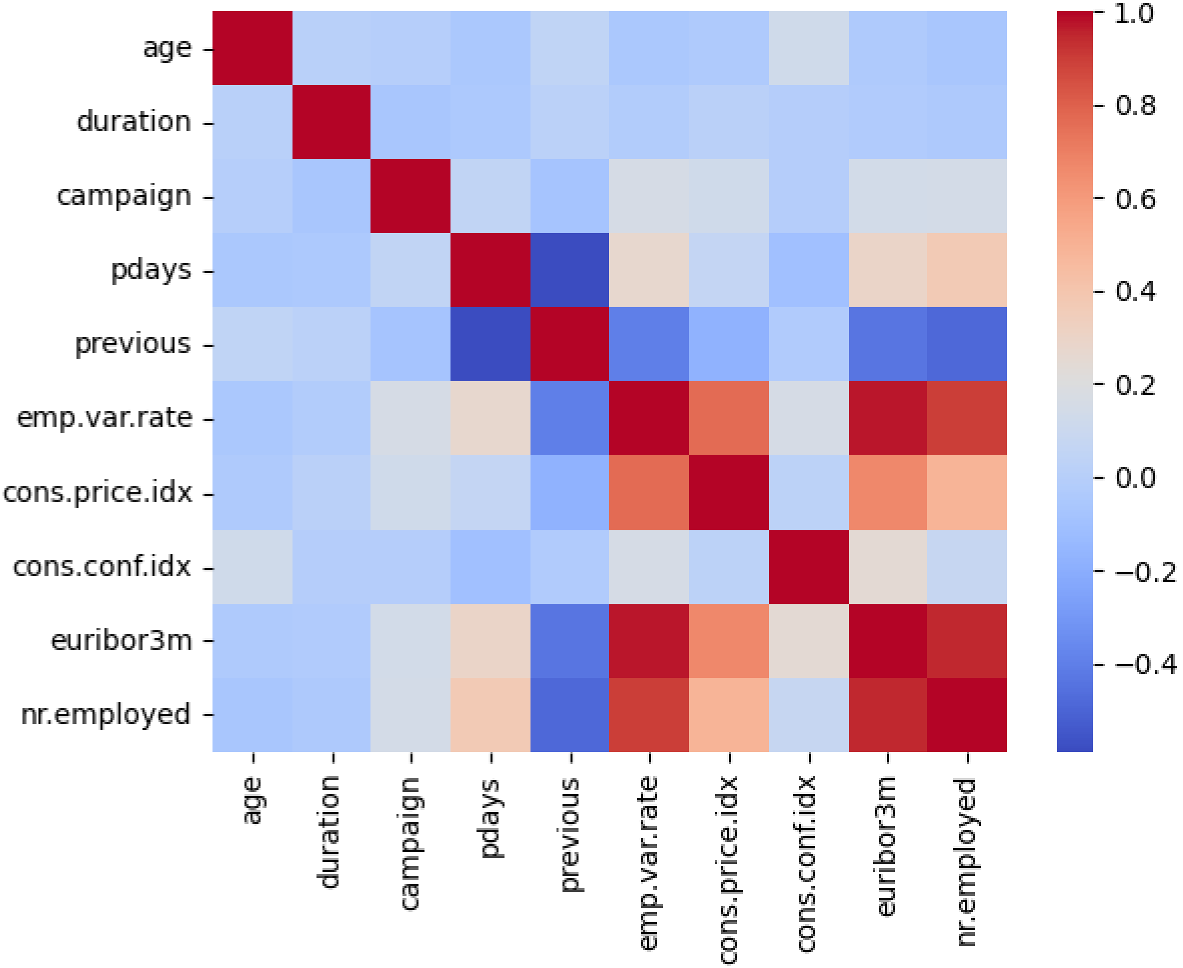
```
<seaborn.axisgrid.JointGrid at 0x7fcba7752e80>
```



So this means that the duration of the last contact was significantly longer in people in the ages between 20 to 60.

In [38]:
```
numeric_df = df[int_data + float_data]
corr_matrix = numeric_df.corr()
sns.heatmap(corr_matrix, cmap='coolwarm')
```

```
Out[38]: <AxesSubplot:>
```



We can see that the data features in integer values are not really correlated whereas the other features in floating point numbers have higher/stronger correlation to each other. It should not always be the case but variables stored as integers often represent categorical or discrete variables with a limited number of values, whereas variables stored as floating point numbers often represent continuous variables with a wider range of values. As a result, integer variables are less likely to be correlated with each other, whereas floating point variables are more likely to be correlated with each other.

```
In [149…
df_encoded = pd.get_dummies(df, columns = obj_data)
df_encoded
```

Out[149]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...	day_of_week_fri	d
0	56	261	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	
2	37	226	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	
3	40	151	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	
4	56	307	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	
6	59	139	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	334	1	999	0	-1.1	94.767	-50.8	1.028	4963.6	...	1	
41184	46	383	1	999	0	-1.1	94.767	-50.8	1.028	4963.6	...	1	
41185	56	189	2	999	0	-1.1	94.767	-50.8	1.028	4963.6	...	1	
41186	44	442	1	999	0	-1.1	94.767	-50.8	1.028	4963.6	...	1	
41187	74	239	3	999	1	-1.1	94.767	-50.8	1.028	4963.6	...	1	

30488 rows × 59 columns

In [150…

```python
from sklearn.model_selection import train_test_split
```

In [151…

```python
X_train, X_test, y_train, y_test = train_test_split(df_encoded, df["y"], test_size = 0.2, random_state = 42)
```

In [152…

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import plot_confusion_matrix, plot_roc_curve
```

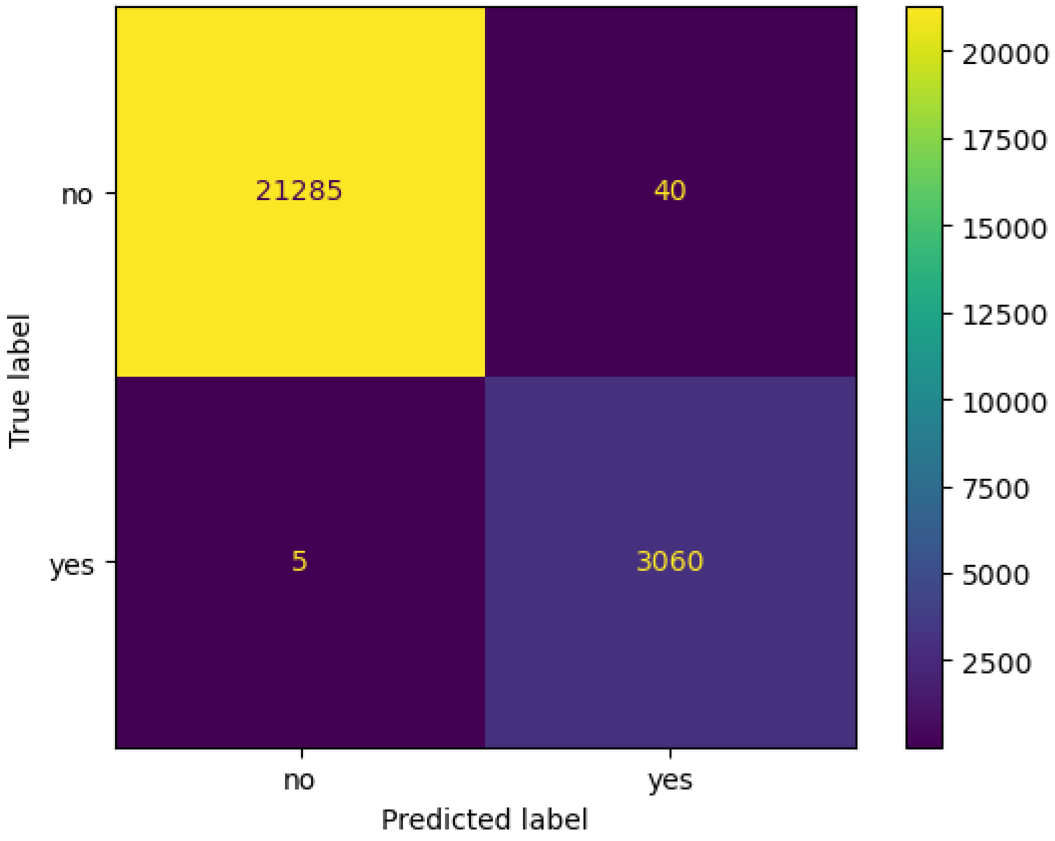
In [153…

```python
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
plot_confusion_matrix(lr, X_train, y_train)
```

```
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plo
t_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Us
e one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

```
Out[153]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fcba7762b50>
```



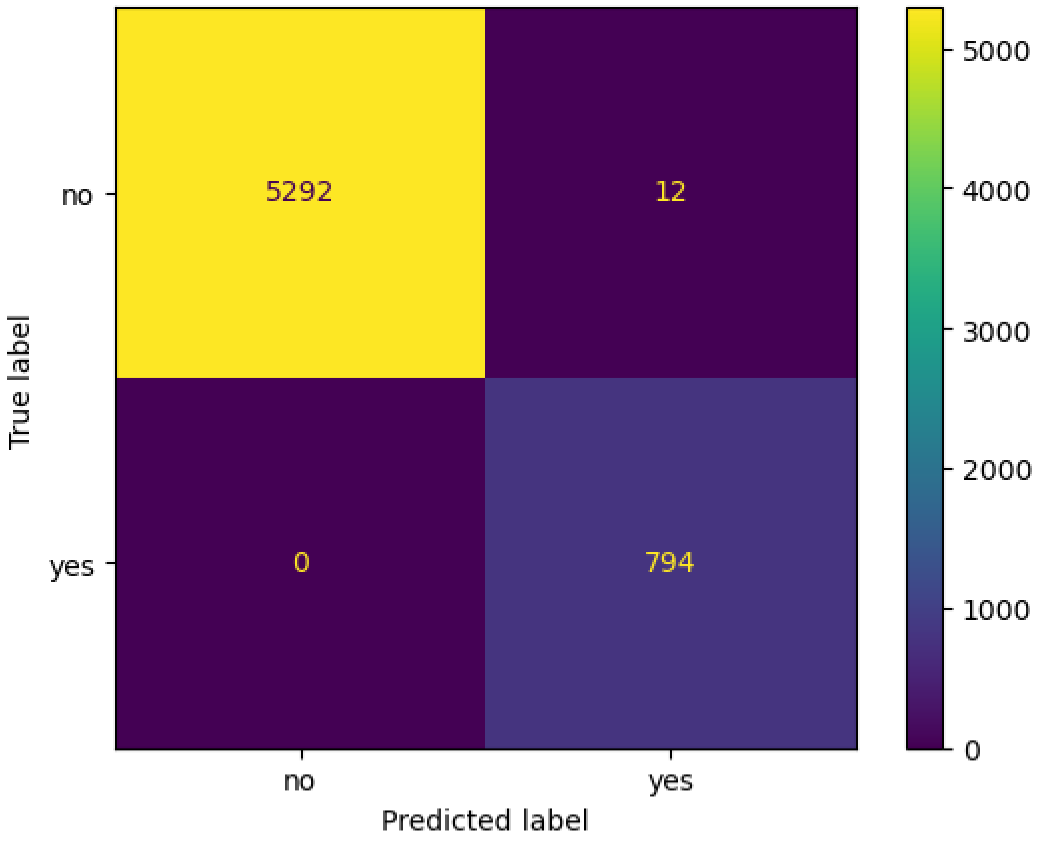
In [154…

```
plot_confusion_matrix(lr, X_test, y_test)
```

```
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

Out[154]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fcb8c90a790>
```

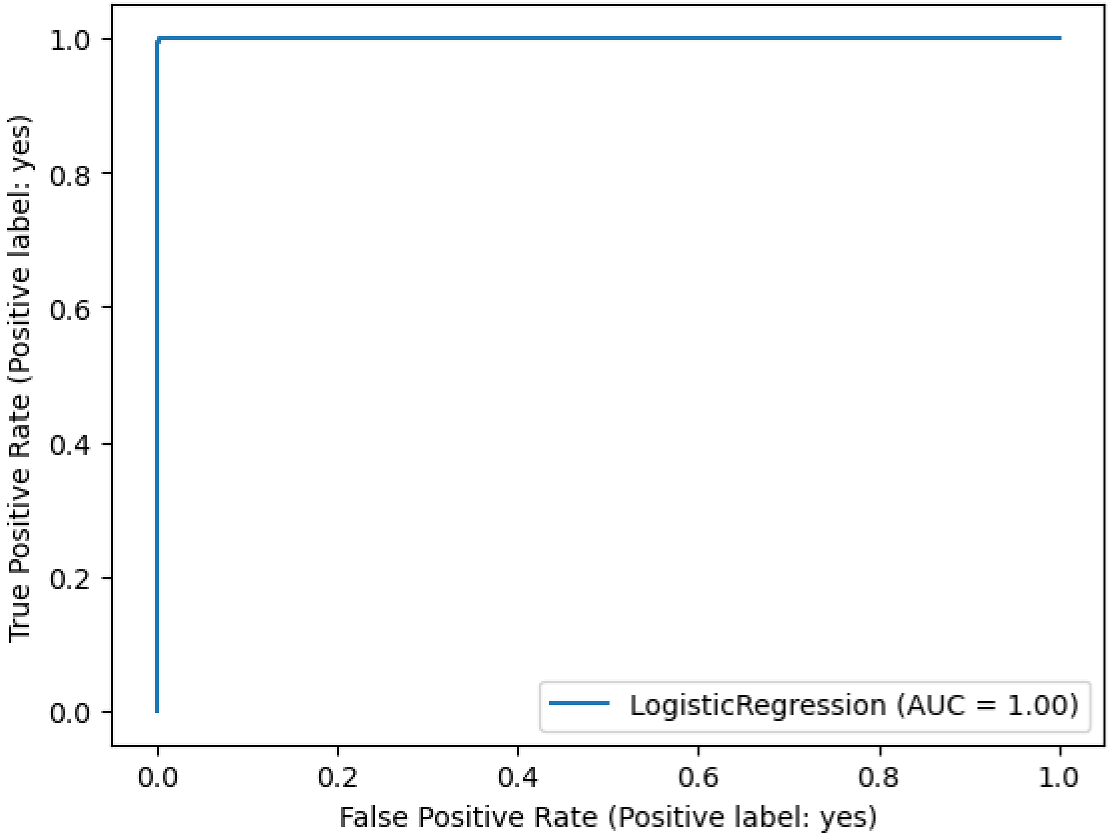


In [155…

```
plot_roc_curve(lr, X_train, y_train)
```

```
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: :meth:`sklearn.metric.RocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.from_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

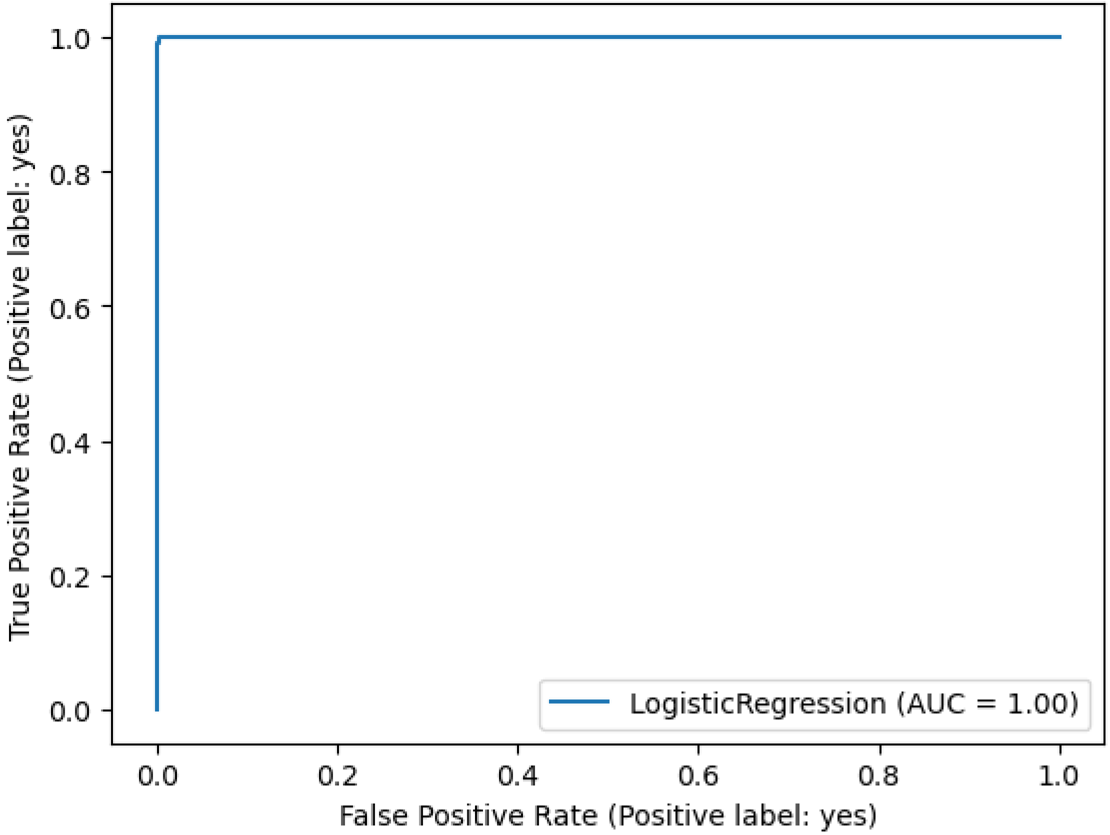
Out[155]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7fcb8c8e88e0>
```



In [156…
```
plot_roc_curve(lr, X_test, y_test)
```

```
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plo
t_roc_curve is deprecated; Function :func:`plot_roc_curve` is deprecated in 1.0 and will be removed in 1.2. Use one of
the class methods: :meth:`sklearn.metric.RocCurveDisplay.from_predictions` or :meth:`sklearn.metric.RocCurveDisplay.fr
om_estimator`.
  warnings.warn(msg, category=FutureWarning)
```

Out[156]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7fcb8c6e3130>
```



In [157…

```
from sklearn.metrics import classification_report
```

In [158…

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

```
              precision    recall  f1-score   support

          no       1.00      1.00      1.00      5304
         yes       0.99      1.00      0.99       794

    accuracy                           1.00      6098
   macro avg       0.99      1.00      1.00      6098
weighted avg       1.00      1.00      1.00      6098
```

In [181…

```
# Print the coefficients of the model
print('Coefficients:\n', lr.coef_)
```

```python
print("\n")

# Print the intercept of the model
print('Intercept:', lr.intercept_)
print("\n")

# Print the probability estimates for the testing set
proba = lr.predict_proba(X_test)
print('Probability estimates:\n', proba)
print("\n")
```

```
Coefficients:
 [[ 5.02666309e-02  3.42533357e-03  3.33118424e-01 -2.61950195e-04
  -6.75275101e-01 -2.91823623e-01  1.25953528e+00 -3.87534314e-01
  -3.31815983e-01 -2.71094949e-02  2.88598488e-01 -4.99956210e-01
  -4.64412780e-02 -1.03721484e-02  1.28550865e-02  1.84193828e-01
   2.86657020e-02 -1.45190166e-01  1.02846555e-01  6.93050931e-02
   2.45862209e-02 -5.34533420e-02 -2.33566122e-01  2.96110636e-01
  -8.82689967e-02 -8.42519493e-02 -2.69943002e-01 -1.01856382e-01
   4.81758634e-03  2.00928662e-02  5.28501049e-01  9.25236852e-03
  -1.61196902e-04  5.96638407e-02 -5.05726691e-02  2.68884330e-02
  -1.77972613e-02  3.95493115e-01 -3.86401944e-01  1.66834783e-01
   2.29840833e-01 -5.26304657e-03  2.89285894e-01  2.44262482e-01
   4.82115189e-01 -1.34886663e+00 -1.37552817e-01  9.14032469e-02
  -2.96875835e-03 -1.06256233e-01 -1.71129103e-01  8.73181867e-02
   4.66833017e-02  1.52475020e-01 -6.04827602e-01  5.37145845e-01
   7.67729294e-02 -8.26867642e+00  8.27776759e+00]]


Intercept: [0.00910516]


Probability estimates:
 [[9.99999325e-01 6.75092519e-07]
 [9.99997564e-01 2.43550265e-06]
 [4.15904429e-03 9.95840956e-01]
 ...
 [9.99999913e-01 8.68772516e-08]
 [9.99999944e-01 5.58018477e-08]
 [9.99494492e-01 5.05508232e-04]]
```

In [182…

```
linear_combination = np.dot(X_test, lr.coef_.T) + lr.intercept_
# Apply the sigmoid function to the linear combination
sigmoid = 1 / (1 + np.exp(-linear_combination))
# Print the probability estimates from the manual calculation
print('Probability estimates from manual calculation: \n', sigmoid)
```

```
Probability estimates from manual calculation:
 [[6.75092519e-07]
 [2.43550265e-06]
 [9.95840956e-01]
 ...
 [8.68772516e-08]
 [5.58018477e-08]
 [5.05508232e-04]]
```

The probability estimates obtained through manual calculation are consistent with those from logistic regression. While both logistic regression and linear regression are methods for understanding how an outcome is related to one or more factors, they differ in their application. Logistic regression is used when the outcome is binary, resulting in a "yes" or "no" answer. In contrast, linear regression is used when the outcome is a number. Logistic regression calculates the probability of an outcome occurring based on the factor values, providing an answer between 0 and 1. On the other hand, linear regression calculates the predicted outcome value based on factor values, providing a continuous answer. A logistic regression model with 100% accuracy may be overfitting the training data, meaning that it has memorized the data instead of detecting the underlying patterns. To identify overfitting, the model's accuracy on new and unseen data (test data) should be compared to that on the training data. If the accuracy of the model on test data is considerably lower than its accuracy on the training data, it may be overfitting.

In [164…

```
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.datasets import make_classification

# Define the hyperparameters to test
param_grid = {'C': [0.1, 1, 10], 'penalty': ['l1', 'l2']}

# Perform grid search using cross-validation to find the best hyperparameters
grid_search = GridSearchCV(lr, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Print the best hyperparameters found by grid search
print('Best hyperparameters:', grid_search.best_params_)

# Evaluate the model using cross-validation with the best hyperparameters
cross_val_scores = cross_val_score(grid_search.best_estimator_, X_test, y_test, cv=5)
```

```
print('Cross-validation scores:', cross_val_scores)
print('Mean cross-validation score:', cross_val_scores.mean())
```

```
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_validation.py:378: FitFailedWarnin
g:
15 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
15 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_validation.py", line 686, i
n _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py", line 1091, in fi
t
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py", line 61, in _che
ck_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/model_selection/_search.py:953: UserWarning: One or
more of the test scores are non-finite: [       nan 0.97269373        nan 0.96793768        nan 0.98405084]
  warnings.warn(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
Best hyperparameters: {'C': 10, 'penalty': 'l2'}
Cross-validation scores: [0.97622951 1.         1.         1.         0.99917966]
Mean cross-validation score: 0.9950818327304024
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/Users/jasonjin/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:444: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Cross-validation and grid search techniques help in enhancing the logistic regression model's performance by tuning its hyperparameters. Through the optimization process, the best set of hyperparameters is determined, allowing the model to fit the data more effectively and generalize well to new, unseen data. Ultimately, the outcome is an improvement in the accuracy or other performance metric of the model, providing a significant lift in the model's performance.

```
In [ ]:
```