```
#To implement Logistic Regression using any inbuild and external data set.
```

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss, roc_auc_score, recall_score, precision_score, average_precision_score, f1_score, classification
```

```
df = pd.read_csv('data.csv', na_values='?')
```

```
df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num       '],
      dtype='object')
```

```
df = df.rename(columns={'num       ': 'target'})

df['target'].value_counts(dropna=False)
```

```
0    188
1    106
Name: target, dtype: int64
```

```
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	28	1	2	130.0	132.0	0.0	2.0	185.0	0.0	0.0	NaN	NaN	NaN	0
1	29	1	2	120.0	243.0	0.0	0.0	160.0	0.0	0.0	NaN	NaN	NaN	0
2	29	1	2	140.0	NaN	0.0	0.0	170.0	0.0	0.0	NaN	NaN	NaN	0
3	30	0	1	170.0	237.0	0.0	1.0	170.0	0.0	0.0	NaN	NaN	6.0	0
4	31	0	2	100.0	219.0	0.0	1.0	150.0	0.0	0.0	NaN	NaN	NaN	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
289	52	1	4	160.0	331.0	0.0	0.0	94.0	1.0	2.5	NaN	NaN	NaN	1
290	54	0	3	130.0	294.0	0.0	1.0	100.0	1.0	0.0	2.0	NaN	NaN	1
291	56	1	4	155.0	342.0	1.0	0.0	150.0	1.0	3.0	2.0	NaN	NaN	1
292	58	0	2	180.0	393.0	0.0	0.0	110.0	1.0	1.0	2.0	NaN	7.0	1
293	65	1	4	130.0	275.0	0.0	1.0	115.0	1.0	1.0	2.0	NaN	NaN	1

294 rows × 14 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       294 non-null    int64  
 1   sex       294 non-null    int64  
 2   cp        294 non-null    int64  
 3   trestbps  293 non-null    float64
 4   chol      271 non-null    float64
 5   fbs       286 non-null    float64
 6   restecg   293 non-null    float64
 7   thalach   293 non-null    float64
 8   exang     293 non-null    float64
 9   oldpeak   294 non-null    float64
 10  slope     104 non-null    float64
 11  ca        3 non-null      float64
 12  thal      28 non-null     float64
 13  target    294 non-null    int64  
dtypes: float64(10), int64(4)
memory usage: 32.3 KB
```

```
df = df.drop(['slope', 'ca', 'thal'], axis=1)

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 261 entries, 0 to 293
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       261 non-null    int64  
 1   sex       261 non-null    int64  
 2   cp        261 non-null    int64  
 3   trestbps  261 non-null    float64
 4   chol      261 non-null    float64
 5   fbs       261 non-null    float64
 6   restecg   261 non-null    float64
 7   thalach   261 non-null    float64
 8   exang     261 non-null    float64
 9   oldpeak   261 non-null    float64
 10  target    261 non-null    int64  
dtypes: float64(7), int64(4)
memory usage: 24.5 KB
```

```
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
0	28	1	2	130.0	132.0	0.0	2.0	185.0	0.0	0.0	0
1	29	1	2	120.0	243.0	0.0	0.0	160.0	0.0	0.0	0
3	30	0	1	170.0	237.0	0.0	1.0	170.0	0.0	0.0	0
4	31	0	2	100.0	219.0	0.0	1.0	150.0	0.0	0.0	0
5	32	0	2	105.0	198.0	0.0	0.0	165.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...
289	52	1	4	160.0	331.0	0.0	0.0	94.0	1.0	2.5	1
290	54	0	3	130.0	294.0	0.0	1.0	100.0	1.0	0.0	1
291	56	1	4	155.0	342.0	1.0	0.0	150.0	1.0	3.0	1
292	58	0	2	180.0	393.0	0.0	0.0	110.0	1.0	1.0	1
293	65	1	4	130.0	275.0	0.0	1.0	115.0	1.0	1.0	1

261 rows × 11 columns

```
#Transform the Categorical Variables: Creating Dummy Variables
df['cp'].value_counts(dropna=False)

df['restecg'].value_counts(dropna=False)
```

```
0.0    208
1.0     47
2.0      6
Name: restecg, dtype: int64
```

```
df = pd.get_dummies(df, columns=['cp', 'restecg'], drop_first=True)

df
```

	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	target	cp_2	cp_3	cp_4	restecg_1.0	restecg_2.0
0	28	1	130.0	132.0	0.0	185.0	0.0	0.0	0	1	0	0	0	1
1	29	1	120.0	243.0	0.0	160.0	0.0	0.0	0	1	0	0	0	0
3	30	0	170.0	237.0	0.0	170.0	0.0	0.0	0	0	0	0	1	0

```
numeric_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
cat_cols = list(set(df.columns) - set(numeric_cols) - {'target'})
cat_cols.sort()

print(numeric_cols)
print(cat_cols)
```

```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
['cp_2', 'cp_3', 'cp_4', 'exang', 'fbs', 'restecg_1.0', 'restecg_2.0', 'sex']
```

```
random_seed = 888
df_train, df_test = train_test_split(df, test_size=0.2, random_state=random_seed, stratify=df['target'])



print(df_train.shape)
print(df_test.shape)
print()
print(df_train['target'].value_counts(normalize=True))
print()
print(df_test['target'].value_counts(normalize=True))
```

```
(208, 14)
(53, 14)

0    0.625
1    0.375
Name: target, dtype: float64

0    0.622642
1    0.377358
Name: target, dtype: float64
```

```
#Transform the Numerical Variables: Scaling
scaler = StandardScaler()
scaler.fit(df_train[numeric_cols])

def get_features_and_target_arrays(df, numeric_cols, cat_cols, scaler):
    X_numeric_scaled = scaler.transform(df[numeric_cols])
    X_categorical = df[cat_cols].to_numpy()
    X = np.hstack((X_categorical, X_numeric_scaled))
    y = df['target']
    return X, y

X, y = get_features_and_target_arrays(df_train, numeric_cols, cat_cols, scaler)
```

```
#Fit the Logistic Regression Model
clf = LogisticRegression(penalty='none') # logistic regression with no penalty term in the cost function.

clf.fit(X, y)
```

```
LogisticRegression(penalty='none')
```

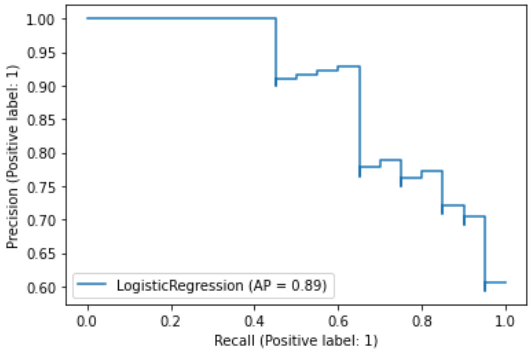
```
#Evaluate the Model
X_test, y_test = get_features_and_target_arrays(df_test, numeric_cols, cat_cols, scaler)
```

```
plot_roc_curve(clf, X_test, y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_roc_curve is deprecated; F
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f785d055fd0>
```

```
plot_precision_recall_curve(clf, X_test, y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_precision_recall_curve is
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at 0x7f785cf871d0>
```



```
test_prob = clf.predict_proba(X_test)[:, 1]
test_pred = clf.predict(X_test)
```

```
print('Log loss = {:.5f}'.format(log_loss(y_test, test_prob)))
print('AUC = {:.5f}'.format(roc_auc_score(y_test, test_prob)))
print('Average Precision = {:.5f}'.format(average_precision_score(y_test, test_prob)))
print('\nUsing 0.5 as threshold:')
print('Accuracy = {:.5f}'.format(accuracy_score(y_test, test_pred)))
print('Precision = {:.5f}'.format(precision_score(y_test, test_pred)))
print('Recall = {:.5f}'.format(recall_score(y_test, test_pred)))
print('F1 score = {:.5f}'.format(f1_score(y_test, test_pred)))

print('\nClassification Report')
print(classification_report(y_test, test_pred))
```

```
Log loss = 0.35613
AUC = 0.92424
Average Precision = 0.89045

Using 0.5 as threshold:
Accuracy = 0.83019
Precision = 0.76190
Recall = 0.80000
F1 score = 0.78049

Classification Report
              precision    recall  f1-score   support

           0       0.88      0.85      0.86        33
           1       0.76      0.80      0.78        20

    accuracy                           0.83        53
   macro avg       0.82      0.82      0.82        53
weighted avg       0.83      0.83      0.83        53
```

```
print('Confusion Matrix')
plot_confusion_matrix(clf, X_test, y_test)
```

```
Confusion Matrix
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f785ca60e90>
```

```
#Interpret the Results
coefficients = np.hstack((clf.intercept_, clf.coef_[0]))
pd.DataFrame(data={'variable': ['intercept'] + cat_cols + numeric_cols, 'coefficient': coefficients})
```

	variable	coefficient
0	intercept	-0.178340
1	cp_2	-2.895253
2	cp_3	-1.808676
3	cp_4	-0.830942
4	exang	0.514580
5	fbs	1.514143
6	restecg_1.0	-0.638990
7	restecg_2.0	-0.429625
8	sex	1.290292
9	age	0.059633
10	trestbps	-0.013132
11	chol	0.345501
12	thalach	-0.285511
13	oldpeak	1.231252

```
pd.DataFrame(data={'variable': numeric_cols, 'unit': np.sqrt(scaler.var_)})
```

	variable	unit
0	age	7.909365
1	trestbps	18.039942
2	chol	63.470764
3	thalach	24.071915
4	oldpeak	0.891801