Hands-on: Logistic Regression

Demo- Logistic Regression

- We will be using the Heart Disease Dataset, with 303 rows and 13 attributes with a target column.
- In this example we will build a classifier to predict if a patient has heart disease of not.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	target
2	63	1	3	145	233	1	0	150	0	2.3	0	0	1
3	37	1	2	130	250	0	1	187	0	3.5	0	0	1
4	41	0	1	130	204	0	0	172	0	1.4	2	0	1
5	56	1	1	120	236	0	1	178	0	0.8	2	0	1
6	57	0	0	120	354	0	1	163	1	0.6	2	0	1
7	57	1	0	140	192	0	1	148	0	0.4	1	0	1
8	56	0	1	140	294	0	0	153	0	1.3	1	0	1
9	44	1	1	120	263	0	1	173	0	0	2	0	1
10	52	1	2	172	199	1	1	162	0	0.5	2	0	1
11	57	1	2	150	168	0	1	174	0	1.6	2	0	1
12	54	1	0	140	239	0	1	160	0	1.2	2	0	1
13	48	0	2	130	275	0	1	139	0	0.2	2	0	1
14	49	1	1	130	266	0	1	171	0	0.6	2	0	1
15	64	1	3	110	211	0	0	144	1	1.8	1	0	1
16	58	0	3	150	283	1	0	162	0	1	2	0	1
17	50	0	2	120	219	0	1	158	0	1.6	1	0	1
18	58	0	2	120	340	0	1	172	0	0	2	0	1
19	66	0	3	150	226	0	1	114	0	2.6	0	0	1
20	43	1	0	150	247	0	1	171	0	1.5	2	0	1

Logistic Regression

Loading the Heart dataset:

1

```
In [1]: import pandas as pd
        dataset = pd.read_csv('heart1.csv')
```

Having a glance at the dataset:

2

```
In [2]: dataset
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	1

Having a look at the shape of the dataset:

3

```
In [3]: print(dataset.shape)
```

```
(303, 13)
```

Logistic Regression

Visualizing the change in the variables

1

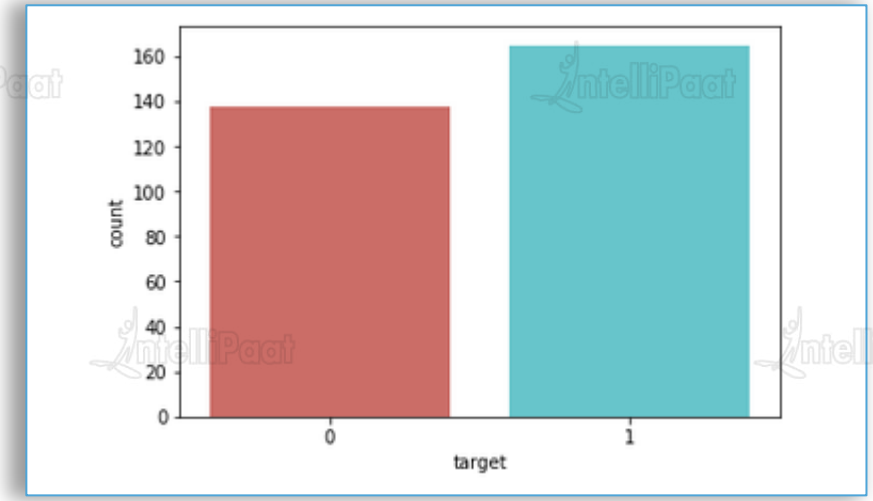
```
In [4]: dataset['target'].value_counts()
```

```
1    165
0    138
Name: target, dtype: int64
```

Visualizing the change in the variables

2

```
In [5]: import matplotlib.pyplot as plt
        import seaborn as sns
        sns.countplot(x= 'target',data = dataset, palette = 'hls')
        plt.show()
```



Logistic Regression

Divide the data into independent and dependent variables:

1

```
In [6]: X = pd.DataFrame(dataset.iloc[:,:-1])
        y = pd.DataFrame(dataset.iloc[:,-1])
```

Split the data into train and test set:

2

```
In [9]: # Import module to split dataset
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Logistic Regression

Train the algorithm:

1

```
In [10]: # Import module for fitting
         from sklearn.linear_model import LogisticRegression
         # Create instance (i.e. object) of LogisticRegression
         logmodel = LogisticRegression()
         logmodel.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

Predicting the test set results:

2

```
In [11]: y_pred = logmodel.predict(X_test)
```

Logistic Regression

Calculating accuracy:

1

```
In [12]: print('Accuracy: %d', (logmodel.score(X_test, y_test)))
```

```
Accuracy: %d 0.7377049180327869
```

Evaluate the model using Confusion Matrix:

2

```
In [14]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.77	0.67	0.71	30
1	0.71	0.81	0.76	31
micro avg	0.74	0.74	0.74	61
macro avg	0.74	0.74	0.74	61
weighted avg	0.74	0.74	0.74	61

Logistic Regression

Evaluating the Algorithm with ROC curve:

```
In [15]: from sklearn.metrics import roc_auc_score
         from sklearn.metrics import roc_curve
         logit_roc_auc = roc_auc_score(y_test, logmodel.predict(X_test))
         fpr, tpr, thresholds = roc_curve(y_test, logmodel.predict_proba(X_test)[:,1])
         plt.figure()
         plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
         plt.plot([0, 1], [0, 1],'r--')
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.05])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('Receiver Operating Characteristic')
         plt.legend(loc="lower right")
         plt.savefig('Log_ROC')
         plt.show()
```