LogisticRegressionForPriceClassification

April 4, 2020

[2]:
```
#using if new data is above median or below median
import pandas as pd
import matplotlib.pyplot as plt
```

[3]:
```
housing_data = pd.read_csv('datasets/housing.csv')
housing_data.sample(5)
```

[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
12749	-121.37	38.62	27.0	1743.0	380.0	
13888	-116.51	34.85	15.0	3149.0	713.0	
16960	-122.30	37.53	43.0	1748.0	366.0	
12801	-121.46	38.62	35.0	3326.0	696.0	
15076	-116.99	32.81	25.0	4436.0	758.0	

	population	households	median_income	median_house_value	\
12749	697.0	368.0	1.6678	166100.0	
13888	1281.0	486.0	2.0000	64700.0	
16960	984.0	371.0	4.5116	337800.0	
12801	2511.0	649.0	1.9871	60900.0	
15076	1997.0	738.0	4.2386	201000.0	

	ocean_proximity
12749	INLAND
13888	INLAND
16960	NEAR OCEAN
12801	INLAND
15076	<1H OCEAN

[4]:
```
housing_data = housing_data.dropna()
```

[5]:
```
housing_data.shape
```

[5]:
```
(20433, 10)
```

[6]:
```
housing_data.loc[housing_data['median_house_value'] == 500001].count()
```

[6]:
```
longitude             958
latitude              958
housing_median_age    958
total_rooms           958
total_bedrooms        958
population            958
households            958
median_income         958
median_house_value    958
ocean_proximity       958
dtype: int64
```

[7]:
```
#drop housing data entry that have median_house_value of 500001
housing_data = housing_data.drop(housing_data.
 ↪loc[housing_data['median_house_value'] == 500001].index)
```

[8]:
```
housing_data.shape
```

[8]:
```
(19475, 10)
```

[9]:
```
housing_data.head()
```

[9]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
0	-122.23	37.88	41.0	880.0	129.0	
1	-122.22	37.86	21.0	7099.0	1106.0	
2	-122.24	37.85	52.0	1467.0	190.0	
3	-122.25	37.85	52.0	1274.0	235.0	
4	-122.25	37.85	52.0	1627.0	280.0	

	population	households	median_income	median_house_value	ocean_proximity
0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	496.0	177.0	7.2574	352100.0	NEAR BAY
3	558.0	219.0	5.6431	341300.0	NEAR BAY
4	565.0	259.0	3.8462	342200.0	NEAR BAY

[10]:
```
housing_data['ocean_proximity'].unique()
```

[10]:
```
array(['NEAR BAY', '<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'ISLAND'],
      dtype=object)
```

[12]:
```
housing_data = pd.get_dummies(housing_data, columns=['ocean_proximity'])
```

[13]:
```
#become 14 column after one hot encoding
housing_data.shape
```

[13]:
```
(19475, 14)
```

[14]:
```
housing_data.sample(5)
```

[14]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
3087	-118.87	35.65	33.0	1504.0	325.0	
16844	-122.43	37.63	34.0	4135.0	687.0	
19831	-119.38	36.56	25.0	1180.0	222.0	
15644	-122.42	37.79	52.0	3457.0	1021.0	
4358	-118.37	34.08	52.0	2946.0	695.0	

	population	households	median_income	median_house_value	\
3087	584.0	223.0	3.4792	94600.0	
16844	2154.0	742.0	4.9732	342300.0	
19831	611.0	212.0	2.0729	84700.0	
15644	2286.0	994.0	2.5650	225000.0	
4358	1258.0	650.0	3.9783	374100.0	

	ocean_proximity_<1H OCEAN	ocean_proximity_INLAND	\
3087	0	1	
16844	0	0	
19831	0	1	
15644	0	0	
4358	1	0	

	ocean_proximity_ISLAND	ocean_proximity_NEAR BAY	\
3087	0	0	
16844	0	0	
19831	0	0	
15644	0	1	
4358	0	0	

	ocean_proximity_NEAR OCEAN
3087	0
16844	1
19831	0
15644	0
4358	0

[16]:
```
#calculate median of median_house_value from datasets, we want to set if its␣
 ↪above or below median value
median = housing_data['median_house_value'].median()
median
```

[16]:
```
173800.0
```

[17]:
```
#append above_median columns into the dataset which value is boolean
#if median_house_value's value is above the median, then return yes
housing_data['above_median'] = (housing_data['median_house_value'] - median) > 0
```

[19]:
```
#another column is appearing "above_median"
housing_data.sample(5)
```

[19]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	\
3491	-118.47	34.27	33.0	1549.0	264.0	
17388	-120.45	34.97	25.0	1920.0	380.0	
9438	-119.98	37.43	12.0	2776.0	592.0	
20185	-119.16	34.27	24.0	1824.0	331.0	
3976	-118.62	34.18	25.0	3124.0	468.0	

	population	households	median_income	median_house_value	\
3491	881.0	289.0	5.1408	222900.0	
17388	1434.0	388.0	3.0368	129300.0	
9438	1236.0	489.0	2.5551	105000.0	
20185	1049.0	320.0	5.9181	221100.0	
3976	1241.0	439.0	6.4044	333100.0	

	ocean_proximity_<1H OCEAN	ocean_proximity_INLAND	\
3491	1	0	
17388	1	0	
9438	0	1	
20185	0	0	
3976	1	0	

	ocean_proximity_ISLAND	ocean_proximity_NEAR BAY	\
3491	0	0	
17388	0	0	
9438	0	0	
20185	0	0	
3976	0	0	

	ocean_proximity_NEAR OCEAN	above_median
3491	0	True
17388	0	False
9438	0	False
20185	1	True
3976	0	True

[21]:
```
#X value dropping the median_house_value and above_median as features
#Y value is boolean / binary classification is that above median or not
X = housing_data.drop(['median_house_value', 'above_median'], axis=1)
Y = housing_data['above_median']
```

[22]:
```
X.columns #features
```

[22]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
```

```
       'ocean_proximity_<1H OCEAN', 'ocean_proximity_INLAND',
       'ocean_proximity_ISLAND', 'ocean_proximity_NEAR BAY',
       'ocean_proximity_NEAR OCEAN'],
      dtype='object')
```

[23]:
```
#split train and test
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)
```

[24]:
```
x_train.shape, x_test.shape
```

[24]:
```
((15580, 13), (3895, 13))
```

[25]:
```
y_train.shape, y_test.shape
```

[25]:
```
((15580,), (3895,))
```

[28]:
```
#Start Logistic Regression.
#liblinear is good choice for small datasets and binary classification
from sklearn.linear_model import LogisticRegression
logistic_model = LogisticRegression(solver='liblinear').fit(x_train,y_train)
```

[30]:
```
#Accuracy of output of classifier on training data
print("Training_score :", logistic_model.score(x_train,y_train))
```

```
Training_score : 0.8188703465982028
```

[31]:
```
y_pred = logistic_model.predict(x_test)
```

[38]:
```
#make new dataframe between predicted and actual
df_pred_actual = pd.DataFrame({'predicted': y_pred, 'actual': y_test})
df_pred_actual.head(10)
```

[38]:

	predicted	actual
10477	True	True
7381	False	False
15130	True	False
18738	False	False
12211	True	True
9541	False	False
1335	False	False
10017	False	True
9691	True	False
7973	True	False

[40]:
```
#print test score (accuracy)
from sklearn.metrics import accuracy_score

print("Testing_score : ", accuracy_score(y_test,y_pred))
```

```
Testing_score :  0.8267008985879333
```

[ ]: