logreg-21mis1152

February 27, 2024

21MIS1152 Rajeev Sekar

LOGISTIC REGRESSION

[15]:
```
#import pandas
import pandas as pd
# load dataset
pima = pd.read_csv("diabetes.csv")
pima.head()
```

[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1

DATA PREPROCESSING

[16]:
```
#split dataset in features and target variable
feature_cols = ['Pregnancies', 'Glucose', 'BMI',␣
 ↪'Age','Insulin','BloodPressure','DiabetesPedigreeFunction']
X = pima[feature_cols] # Features
y = pima.Outcome # Target variable
```

[17]:
```
matrix = X.corr()
print(matrix)
```

```
                          Pregnancies   Glucose       BMI       Age   Insulin  \
Pregnancies                  1.000000  0.129459  0.017683  0.544341 -0.073535
Glucose                      0.129459  1.000000  0.221071  0.263514  0.331357
BMI                          0.017683  0.221071  1.000000  0.036242  0.197859
```

```
Age                           0.544341  0.263514  0.036242  1.000000 -0.042163
Insulin                      -0.073535  0.331357  0.197859 -0.042163  1.000000
BloodPressure                 0.141282  0.152590  0.281805  0.239528  0.088933
DiabetesPedigreeFunction     -0.033523  0.137337  0.140647  0.033561  0.185071

                          BloodPressure  DiabetesPedigreeFunction
Pregnancies                    0.141282                 -0.033523
Glucose                        0.152590                  0.137337
BMI                            0.281805                  0.140647
Age                            0.239528                  0.033561
Insulin                        0.088933                  0.185071
BloodPressure                  1.000000                  0.041265
DiabetesPedigreeFunction       0.041265                  1.000000
```

No features are correlated with each other => We can apply logistic regression algorithm

Scaling the features:

[42]:
```
from sklearn.preprocessing import StandardScaler

std_scaler = StandardScaler()

X = std_scaler.fit_transform(X)
```

Splitting train and test sets

[43]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=16)
```

MODEL TRAINING

[44]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
```

MODEL EVALUATION

[45]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.92      0.87       125
```

```
           1       0.81      0.63      0.71        67

    accuracy                           0.82       192
   macro avg       0.81      0.77      0.79       192
weighted avg       0.82      0.82      0.81       192
```

Interpretation:

=> only 82% of patients predicted as having diabetes actually have diabetes

=> 92% of the patients actually having diabetes are predicted correctly

=> Overall, the model predicts 82% of the cases accurately

Displaying the model parameters:

1. The slope values of all features

[46]:
```
logreg.coef_
```

[46]:
```
array([[ 0.30213836,  1.02049564,  0.70136964,  0.15244513, -0.15129797,
        -0.25676181,  0.29104294]])
```

2. The intercept value

[47]:
```
logreg.intercept_
```

[47]:
```
array([-0.83662093])
```

[ ]:
```

```