regression1

November 8, 2022

Aim: Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

Short notes:

Linear regression assumes a linear relationship between the input variable (X) and a single output variable (Y). When there is a single input variable, the method is referred to as a simple linear regression.

In a simple linear regression, we can estimate the coefficients required by the model to make predictions on new data analytically. That is, the line for a simple linear regression model can be written as:

$y = B0 + B1 * x + \in$

where B0 and B1 are the coefficients we must estimate from the training data and is an error term. Once the coefficients are estimated, we can use this equation to predict output values for y conditional on new input examples of x. We use LinearRegression() from sklearn.linear_model.

Multiple Linear Regression: When one variable/column in a dataset is not sufficient to create a good model and make more accurate predictions, we'll use a multiple linear regression model instead of a simple linear regression model. The line equation for the multiple linear regression model is:

$y = \beta 0 + \beta 1X1 + \beta 2X2 + \beta 3X3 + ... + \beta pXp + e$

We use datasets.load_diabetes dataset for this program.

About the dataset.

The diabetes dataset consists of 10 physiological variables (such as age, sex, weight, blood pressure) measure on 442 patients, and an indication of disease progression after one year. Note that the input variables have all been standardised to each have mean 0 and squared length = 1.

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

[ ]:
```
df = datasets.load_diabetes()
df['feature_names']
```

[ ]:
```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

[ ]:
```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
diabetes_X.shape
```

[ ]:
```
(442, 10)
```

[ ]:
```
diabetes_y.shape
```

[ ]:
```
(442,)
```

[ ]:
```
# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis,2]
diabetes_X.shape
```

[ ]:
```
(442, 1)
```

[ ]:
```
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

[ ]:
```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

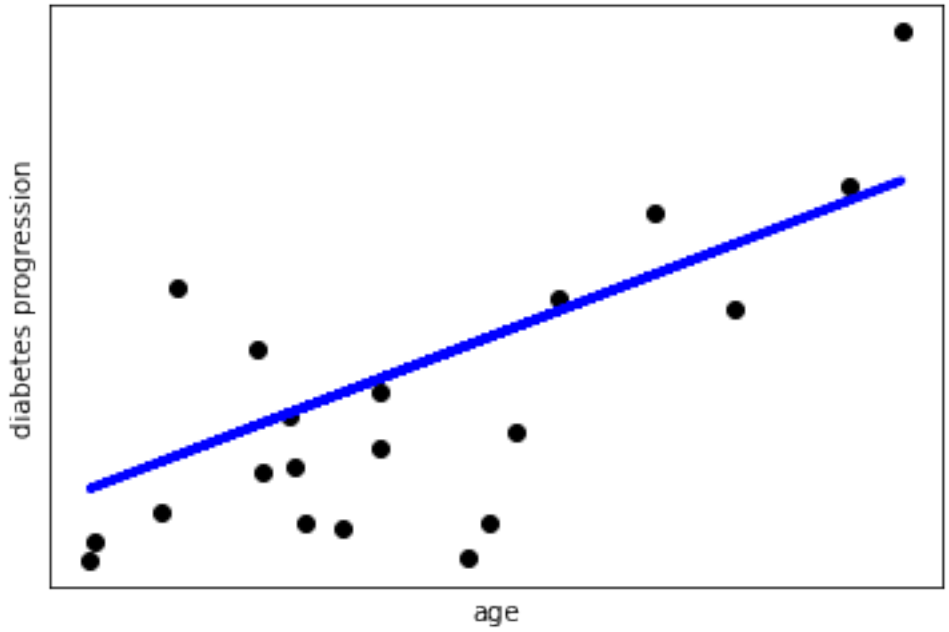
[ ]:
```
# print prediction
print("The predicted value is=", diabetes_y_pred)
# The coefficients
print("Coefficients: \n", regr.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,␣
 ↪diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test,␣
 ↪diabetes_y_pred))
```

```
The predicted value is= [225.9732401  115.74763374 163.27610621 114.73638965
120.80385422
```

```
 158.21988574 236.08568105 121.81509832  99.56772822 123.83758651
 204.73711411  96.53399594 154.17490936 130.91629517  83.3878227
 171.36605897 137.99500384 137.99500384 189.56845268  84.3990668 ]
Coefficients:
 [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

[ ]:
```
# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test, color="black")
plt.plot(diabetes_X_test, diabetes_y_pred, color="blue", linewidth=3)
plt.xlabel("age")
plt.ylabel("diabetes progression")
plt.xticks(())
plt.yticks(())

plt.show()
```



Multiple Regression

[ ]:
```
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
diabetes_X.shape
diabetes_X = diabetes_X[:,[0,2]]
diabetes_X.shape
```

[ ]:
```
(442, 2)
```

[ ]:
```
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
```

[ ]:
```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
```

[ ]:
```
# The coefficients
print("Coefficients: \n", regr.coef_)
print("Intercept: \n", regr.intercept_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,␣
 ↪diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test,␣
 ↪diabetes_y_pred))
```

```
Coefficients:
 [139.20420118 912.45355549]
Intercept:
 152.87670001405584
Mean squared error: 2596.60
Coefficient of determination: 0.46
```

[ ]:
```
x = diabetes_X_test[:, 0]
y = diabetes_X_test[:, 1]
#z = diabetes_X_test[:, 2]
```

The following plots the values in three different angles.

[ ]:
```
plt.style.use('default')

fig = plt.figure(figsize=(12, 4))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
```

```
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
    ax.plot(x, y, diabetes_y_pred, color='k', zorder=15, linestyle='none',␣
 ↪marker='o', alpha=0.5)
    ax.scatter(x.flatten(), y.flatten(), diabetes_y_pred, facecolor=(0,0,0,0),␣
 ↪s=20, edgecolor='#70b3f0')
    ax.set_xlabel('Age', fontsize=12)
    ax.set_ylabel('BMI', fontsize=12)
    ax.set_zlabel('diabetes', fontsize=12)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2_score(diabetes_y_test, diabetes_y_pred),␣
 ↪fontsize=20)

fig.tight_layout()
```

$R^2 = 0.46$