Multiple Linear Regression With scikit-learn

Regression is a statistical method for determining the relationship between features and an outcome variable or result. Machine learning, it's utilized as a method for predictive modeling, in which an algorithm is employed to forecast continuous outcomes. Multiple linear regression, often known as multiple regression, is a statistical method that predicts the result of a response variable by combining numerous explanatory variables. Multiple regression is a variant of linear regression (ordinary least squares) in which just one explanatory variable is used.

Mathematical Imputation:
To improve prediction, more independent factors are combined. The following is the linear relationship between the dependent and independent variables:

$$y = \beta_0 + \beta_1 x_1 +. \beta_2 x_2 + \beta_3 x_3 ....$$

here, y is the dependent variable.

- x1, x2,x3,… are independent variables.
- b0 =intercept of the line.
- b1, b2, … are coefficients.

for a simple linear regression line is of the form :
y = mx+c

for example if we take a simple example, :
feature 1: *TV*
feature 2: *radio*
feature 3: *Newspaper*
output variable: *sales*
Independent variables are the features feature1 , feature 2 and feature 3. Dependent variable is sales. The equation for this problem will be:

y = b0+b1x1+b2x2+b3x3

x1, x2 and x3 are the feature variables.

Evaluate the model with metrics.

The multi-linear regression model is evaluated with mean_squared_error and mean_absolute_error metric. when compared with the mean of the target variable, we'll understand how well our model is predicting. mean_squared_error is the mean of the sum of residuals. mean_absolute_error is the mean of the absolute errors of the model. The less the error, the better the model performance is.

mean absolute error = it's the mean of the sum of the absolute values of residuals.

$$1/n \sum_{i=0}^{n} |y - \overline{y}|$$

mean square error = it's the mean of the sum of the squares of residuals.

$$1/n \sum_{i=0}^{n} (y - \overline{y})\ 2$$

- y= actual value
- y hat = predictions

```python
# importing modules and packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing

from google.colab import files
uploaded = files.upload()
```

```python
# importing data
df = pd.read_csv('Real-estate1.csv')
df.drop('No', inplace = True,axis=1)

print(df.head())
print(df.columns)
```

```python
# plotting a scatterplot
sns.scatterplot(x='X4 number of convenience stores',
                y='Y house price of unit area', data=df)
```

```python
# creating feature variables
X = df.drop('Y house price of unit area',axis= 1)
y = df['Y house price of unit area']
print(X)
print(y)
```

```
# creating train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=101)

```

```
# creating a regression model
model = LinearRegression()
```

```
# fitting the model
model.fit(X_train,y_train)
```

```
# making predictions
predictions = model.predict(X_test)

```

```
# model evaluation
print(
  'mean_squared_error : ', mean_squared_error(y_test, predictions))
print(
  'mean_absolute_error : ', mean_absolute_error(y_test, predictions))
```

Plot Multinomial and One-vs-Rest Logistic Regression in Scikit Learn

Logistic Regression is a popular classification algorithm that is used to predict the probability of a binary or multi-class target variable. In scikit-learn, there are two types of logistic regression algorithms: Multinomial logistic regression and One-vs-Rest logistic regression. Multinomial logistic regression is used when the target variable has more than two classes, while One-vs-Rest logistic regression is used when the target variable has two or more classes.

1. **Multinomial Logistic Regression**: It is a logistic regression algorithm that is used when the target variable has more than two classes. It predicts the probability of each class and selects the class with the highest probability as the predicted class.
2. **One-vs-Rest Logistic Regression**: It is a logistic regression algorithm that is used when the target variable has two or more classes. It trains one logistic regression model for each class, with that class as the positive class and all other classes as the negative class. It predicts the probability of each class and selects the class with the highest probability as the predicted class.

```python
# import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

# load the iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# split the data into training and testing sets
X_train, X_test,\
y_train, y_test = train_test_split(X, y,
                                   test_size=0.2,
                                   random_state=42)

# create a Multinomial logistic regression model
multi_logreg = LogisticRegression(multi_class='multinomial',
                                  solver='lbfgs')
multi_logreg.fit(X_train, y_train)

# create a One-vs-Rest logistic regression model
ovr_logreg = LogisticRegression(multi_class='ovr',
                                solver='liblinear')
ovr_logreg.fit(X_train, y_train)
```

```
# make predictions using the trained models
y_pred_multi = multi_logreg.predict(X_test)
y_pred_ovr = ovr_logreg.predict(X_test)

# evaluate the performance of the models
# using accuracy score and confusion matrix
print('Multinomial logistic regression accuracy:',
      accuracy_score(y_test, y_pred_multi))
print('One-vs-Rest logistic regression accuracy:',
      accuracy_score(y_test, y_pred_ovr))

conf_mat_multi = confusion_matrix(y_test, y_pred_multi)
conf_mat_ovr = confusion_matrix(y_test, y_pred_ovr)
```

```
# plot the confusion matrices
fig, axs = plt.subplots(ncols=2, figsize=(10, 5))
axs[0].imshow(conf_mat_multi, cmap=plt.cm.Blues)
axs[0].set_title('Multinomial logistic regression')
axs[0].set_xlabel('Predicted labels')
axs[0].set_ylabel('True labels')
axs[0].set_xticks(np.arange(len(iris.target_names)))
axs[0].set_xticklabels(iris.target_names)
axs[0].set_yticklabels(iris.target_names)
axs[1].imshow(conf_mat_ovr, cmap=plt.cm.Blues)
axs[1].set_title('One-vs-Rest logistic regression')
axs[1].set_xlabel('Predicted labels')
axs[1].set_ylabel('True labels')
axs[1].set_xticks(np.arange(len(iris.target_names)))
axs[1].set_xticklabels(iris.target_names)
axs[1].set_yticks(np.arange(len(iris.target_names)))
axs[1].set_yticklabels(iris.target_names)
plt.show()
```

Multinomial Logistic Regression Plot

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

# Load the iris dataset
iris = load_iris()

# Extract the features and target
X = iris.data[:, :2]
y = iris.target

# Create an instance of Logistic Regression classifier
clf = LogisticRegression(random_state=0,
                         multi_class='multinomial',
                         solver='newton-cg')

# Fit the model
clf.fit(X, y)

```

```
# Plot the decision boundaries
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                     np.arange(y_min, y_max, .02))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k',
            cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

One-vs-Rest Logistic Regression Plot

For the iris dataset, we will use scikit-learn library in Python to load the dataset and fit the logistic regression model. Then we will use Matplotlib library to plot the decision boundaries which are obtained by using the one-vs-rest Logistic Regression.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

iris = load_iris()

# we only take the first two features for visualization
X = iris.data[:, :2]
y = iris.target

clf = LogisticRegression(random_state=0,
                         multi_class='ovr',
                         solver='liblinear')

clf.fit(X, y)

```

```
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, .02),
                     np.arange(y_min, y_max, .02))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k',
            cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('One-vs-Rest logistic regression')
plt.show()
```