Linear Regression with Python

Reference: https://www.kaggle.com/gopalchettri/usa-housing-machine-learning-linear-regression, https://www.kaggle.com/code

Credit (Image and Good Read): https://towardsdatascience.com/regression-using-sklearn-on-kc-housing-dataset-1ac80ca3d6d4?gi=d6492b1463a1

Your neighbor is a real estate agent and wants some help predicting housing prices for regions in the USA. It would be great if you could somehow create a model for her that allows her to put in a few features of a house and returns back an estimate of what the house would sell for.

She has asked you if you could help her out with your new data science skills. You say yes, and decide that Linear Regression might be a good path to solve this problem!

Your neighbor then gives you some information about a bunch of houses in regions of the United States,it is all in the data set: USA_Housing.csv.

The data contains the following columns:

- 'Avg. Area Income': Avg. Income of residents of the city house is located in.
- 'Avg. Area House Age': Avg Age of Houses in same city
- 'Avg. Area Number of Rooms': Avg Number of Rooms for Houses in same city

- 'Avg. Area Number of Bedrooms': Avg Number of Bedrooms for Houses in same city
- 'Area Population': Population of city house is located in
- 'Price': Price that the house sold at
- 'Address': Address for the house

```
!pip install --upgrade scikit-learn==1.0.2
!pip install --upgrade numpy==1.21.5
```

Let's get started!

Check out the data

We've been able to get some data from your neighbor for housing prices as a csv set, let's get our environment ready with the libraries we'll need and then import the data!

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Check out the Data

```
!rm USA_Housing.csv
!wget https://github.com/davidjohnnn/all_datasets/raw/master/bay/USA_Housing.cs
```

```
USAhousing = pd.read_csv('USA_Housing.csv')
```

```
USAhousing.head()
```

```
USAhousing.info()
```

```
USAhousing.describe()
```

```
USAhousing.columns
```

EDA

Let's create some simple plots to check out the data!

```
sns.pairplot(USAhousing)
```

```
sns.distplot(USAhousing['Price'])
```

```
sns.heatmap(USAhousing.corr())
```

Training a Linear Regression Model

Let's now begin to train out regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable, in this case the Price column. We will toss out the Address column because it only has text info that the linear regression model can't use.

X and y arrays

```
X = USAhousing.drop(['Price','Address'],axis=1)
y = USAhousing['Price']
```

```
X.head(3)
```

```
y.head(3)
```

Train Test Split

Now let's split the data into a training set and a testing set. We will train out model on the training set and then use the test set to evaluate the model.

```
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, randon
```

Creating and Training the Model

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
```

```
lm.fit(X_train,y_train)
```

Model Evaluation

Let's evaluate the model by checking out it's coefficients and how we can interpret them.

```
# print the intercept
print(lm)
print(lm.intercept_)
print(lm.n_features_in_)
print(lm.feature_names_in_)
```

Interpreting the coefficients:

```
coeff_df = pd.DataFrame(lm.coef_,lm.feature_names_in_,columns=['Coefficient'])
coeff_df
```

Does this make sense? Probably not because I made up this data. If you want real data to repeat this sort of analysis, check out the boston dataset:

```
from sklearn.datasets import load_boston
boston = load_boston()
print(boston.DESCR)
boston_df = boston.data
```

Predictions from our Model

Let's grab predictions off our test set and see how well it did!

```
predictions = lm.predict(X_test)
```

```
plt.scatter(y_test,predictions)
```

Residual Histogram

```
sns.distplot((y_test-predictions),bins=50);
```

Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

```
from sklearn import metrics
```

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

This was your first real Machine Learning Project! Congrats on helping your neighbor out! We'll let this end here for now, but go ahead and explore the Boston Dataset mentioned earlier if this particular data set was interesting to you!