The Art of Analytics

EXPLORING VALUE FROM DATA

Ajeet Kushwaha

Lecture#11

Linear Regression Example

- First import all the necessary libraries

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

- train_test_split is a function from sklearn.model_selection module that helps split the dataset into training and testing sets. We import it to split our dataset later.
- LinearRegression is a class from sklearn.linear_model module that represents the linear regression model. We import it to create and train our linear regression model.

- Setup your dataset (Here I'm taking random)

```
X = np.random.rand(100, 1)
y = 2 + 3 * X + np.random.randn(100, 1)
```

- In this step, we generate a dummy dataset for demonstration purposes. X represents the predictor variable, and y represents the target variable.
- We use numpy to create random values for X using the rand function, which generates random numbers from a uniform distribution between 0 and 1.
- We define y by adding noise to a linear relationship with X using the formula 2 + 3 * X + np.random.randn(100, 1). The np.random.randn function generates random numbers from a standard normal distribution, representing the noise added to our linear relationship.
- Creating a dummy dataset helps us demonstrate the implementation of linear regression, but in practice, you would typically load your dataset from a file or a database.

- Splitting the dataset 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)
```

- It is crucial to split the dataset into training and testing sets to evaluate the performance of our model on unseen data.
- The train_test_split function takes the input data (X and y) and splits it into training and testing sets. The parameter test_size=0.2 specifies that we want to allocate 20% of the data for testing, while the remaining 80% will be used for training.
- By splitting the data, we can assess the model's ability to generalize to new, unseen data. The random_state parameter ensures reproducibility by fixing the random seed to 42.

- Creating and training the linear regression model:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

- we create an instance of the LinearRegression class and assign it to the variable model. This class represents the linear regression model in scikit-learn.
- We call the fit method on our model object, passing in the training data (X_train and y_train). This step trains the model by fitting the linear regression line to the training data.
- Training the model involves calculating the optimal values for the model's coefficients (slopes) and intercept by minimizing the sum of squared residuals between the actual and predicted values.

- Getting the coefficient, intercept, and other parameters

```
coef = model.coef_
intercept = model.intercept_
```

- After training the linear regression model, we can access its attributes to obtain useful information.
- The coef_ attribute represents the coefficients (slopes) of the linear regression line. In our case, since we have a single predictor variable, coef_ will be a 1D array with a single value.
- The intercept_ attribute represents the y-intercept of the linear regression line.
- These attributes provide insights into the relationship between the predictor variable and the target variable.

Practice Material

Click Me

Thank You

IN CASE OF ANY DOUBT, FEEL FREE TO ASK ME