Multivariate K-Nearest Neighbors: Takeaways

Syntax

- Displaying the number of non-null values in the columns of a DataFrame:

```
dc_listings.info()
```

- Removing rows from a DataFrame that contain a missing value:

```
dc_listings.dropna(axis=0, inplace=True)
```

- Normalizing a column using pandas:

```
first_transform = dc_listings['maximum_nights'] - dc_listings['maximum_nights'].mean()
normalized_col = first_transform / first_transform.std()
```

- Normalizing a DataFrame using pandas:

```
normalized_listings = (dc_listings - dc_listings.mean()) / (dc_listings.std())
```

- Calculating Euclidean distance using SciPy:

```
from scipy.spatial import distance
first_listing = [-0.596544, -0.439151]
second_listing = [-0.596544, 0.412923]
dist = distance.euclidean(first_listing, second_listing)
```

- Using the KNeighborsRegressor to instantiate an empty model for K-Nearest Neighbors:

```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor()
```

- Using the fit method to fit the K-Nearest Neighbors model to the data:

```
train_df = normalized_listings.iloc[0:2792]
test_df = normalized_listings.iloc[2792:]
train_features = train_df[['accommodates', 'bathrooms']]
train_target = train_df['price']
knn.fit(train_features, train_target)
```

- Using the predict method to make predictions on the test set:

```
predictions = knn.predict(test_df[['accommodates', 'bathrooms']])
```

- Calculating MSE using scikit-learn:

```
from sklearn.metrics import mean_squared_error
two_features_mse = mean_squared_error(test_df['price'], predictions)
```

Concepts

- To reduce the RMSE value during validation and improve accuracy, you can:
    - Select the relevant attributes a model uses. When selecting attributes, you want to make sure you're not working with a column that doesn't have continuous values. The process of selecting features to use in a model is known as feature selection.

    - Increase the value of `k` in our algorithm.
- We can normalize the columns to prevent any single value having too much of an impact on distance. Normalizing the values to a standard normal distribution preserves the distribution while aligning the scales. Let `x` be a value in a specific column, μ be the mean of all values within a single column, and σ be the standard deviation of the values within a single column, then the mathematical formula to normalize the values is as follows:
$$x = \frac{x-\mu}{\sigma}$$
- The `distance.euclidean()` function from `scipy.spatial` expects:
    - Both of the vectors to be represented using a list-like object (Python list, NumPy array, or pandas Series).
    - Both of the vectors must be 1-dimensional and have the same number of elements.
- The scikit-learn library is the most popular machine learning library in Python. Scikit-learn contains functions for all of the major machine learning algorithms implemented as a separate class. The workflow consists of four main steps:
    - Instantiate the specific machine learning model you want to use.
    - Fit the model to the training data.
    - Use the model to make predictions.
    - Evaluate the accuracy of the predictions.
- One main class of machine learning models is known as a regression model, which predicts numerical value. The other main class of machine learning models is called classification, which is used when we're trying to predict a label from a fixed set of labels.
- The fit method accepts list-like objects while the predict method accepts matrix like objects.
- The `mean_squared_error()` function takes in two inputs:
    - A list-like object representing the actual values.
    - A list like object representing the predicted values using the model.

Resources

- Scikit-learn library
- K-Neighbors Regressor