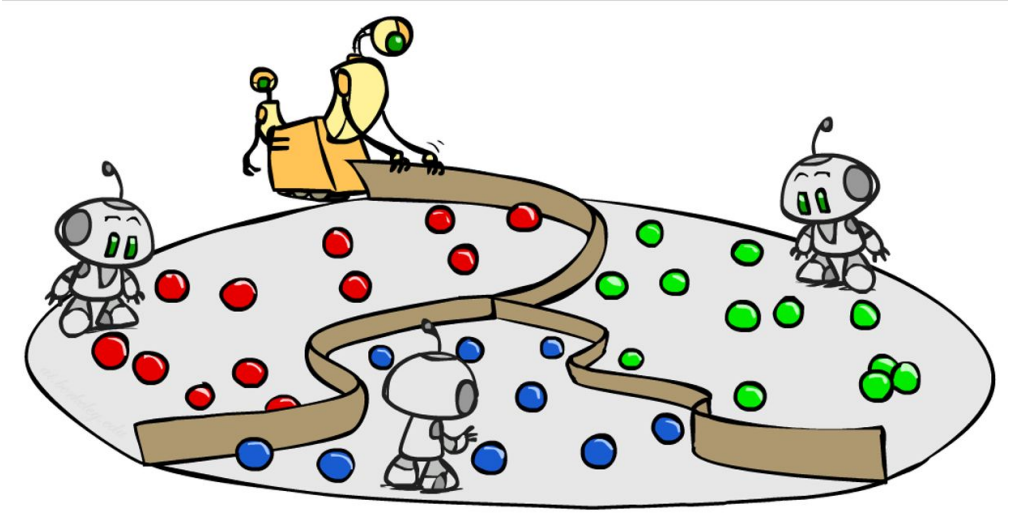
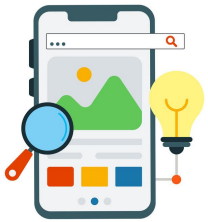
CS-ELEC1A: Advanced Intelligent Systems

Lab Exercise #1: Train Your Own Linear Regression



Problem Context



Context: **Housing in India** varies from palaces of erstwhile maharajas to modern apartment buildings in big cities to tiny huts in far-flung villages. There has been tremendous growth in India's housing sector as incomes have risen.

Suppose you are a Data Scientist from a **real-estate company** that's providing leases and letting for your customers. Your **objective is to create a model that identifies the renting price of a property** based on information from housing data.

Your model will be used to competitively price your company's properties against other offerings in the market so it has to be as good as it can be.

Provided Dataset

BHK: Number of Bedrooms, Hall, Kitchen.

Rent: Rent of the Houses/Apartments/Flats.

Size: Size of the Houses/Apartments/Flats in Square Feet.

Floor: Houses/Apartments/Flats situated in which Floor and Total Number of Floors (Example: Ground out of 2, 3 out of 5, etc.)

Area Type: Size of the Houses/Apartments/Flats calculated on either Super Area or Carpet Area or Build Area.

Area Locality: Locality of the Houses/Apartments/Flats.

City: City where the Houses/Apartments/Flats are Located.

Furnishing Status: Furnishing Status of the Houses/Apartments/Flats, either it is Furnished or Semi-Furnished or Unfurnished.

Tenant Preferred: Type of Tenant Preferred by the Owner or Agent.

Bathroom: Number of Bathrooms.

Point of Contact: Whom should you contact for more information regarding the Houses/Apartments/Flats.

Step #1: Installing the Necessary Libraries

Install Pandas, Numpy, Scikit-Learn, Seaborn, and Matplotlib.

- **Numpy** is a **Numerical Python** package that allows us to easily do matrix multiplications, and other relevant numeric operations. Mostly used for scientific computing
- **Pandas** is a **Python Data Analysis** toolkit that allows us to easily manipulate data using easy-to-use data structures. It is built on top of Numpy
- **Scikit-Learn** is a library for **machine learning** and **predictive analysis** in Python
- **Matplotlib** and **Seaborn** is a library for **plotting and visualization**

Step #1: Installing the Necessary Libraries

Command:

```
pip install pandas
pip install numpy
pip install scikit-learn
pip install matplotlib
pip install seaborn
```

Step #2: Downloading the Dataset

Retrieve the dataset from https://drive.google.com/file/d/1PGiyGKAZOBf_bp0jQ3ThX1jIWmNAIvRi/view?usp=drive_link

Step #3: Import and Checking the Dataset

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

### Loading the Dataset
data = pd.read_csv("House_Rent_Dataset.csv")
data.head(10)

data.info()

data.isnull().sum()
```

Step #4: Simple Exploratory Data Analysis

```
plt.bar(data['BHK'].value_counts().index, data['BHK'].value_counts().values)

sns.distplot(data['Rent'])

sns.distplot(data['Size'])

data['Floor'].value_counts()

data['Furnishing Status'].value_counts()

data['Area Locality'].value_counts()
```

Step #5: Preprocessing (Encoding)

```
### Conversion of Categorical Variable to One-Hot Encoding
data = data[['BHK', 'Bathroom', 'Furnishing Status', 'Rent']]

def one_hot_encode(data, column):
    encoded = pd.get_dummies(data[column], drop_first= True)
    data = data.drop(column, axis = 1)
    data = data.join(encoded)
    return data

data = one_hot_encode(data, 'Furnishing Status')
data
```

Step #5: Preprocessing (Encoding)

```
### Conversion of Categorical Variable to One-Hot Encoding
data = data[['BHK', 'Bathroom', 'Furnishing Status', 'Rent']]

def one_hot_encode(data, column):
    encoded = pd.get_dummies(data[column], drop_first= True)
    data = data.drop(column, axis = 1)
    data = data.join(encoded)
    return data

data = one_hot_encode(data, 'Furnishing Status')
data
```

Step #5: Preprocessing (Training and Test Split)

```
X = data.drop('Rent', axis= 1)
y = data['Rent']

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, random_state = 42)
```

Step #5: Preprocessing (Standardization)

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Step #6: Modelling

```
from sklearn import linear_model
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
model.coef_
```

Step #7: Evaluation

```
### Quantitative Evaluation
y_preds = model.predict(X_test)

from sklearn.metrics import mean_squared_error, r2_score

# The coefficients
print("Coefficients: \n", model.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_preds))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test, y_preds))
```

Step #7: Evaluation

```
### Qualitative Evaluation
sample_data = X.iloc[0]
sample_data

sample_data_standardized = sc.transform(X.iloc[0].values.reshape(1,-1))

model_rent_forecast = model.predict(sample_data_standardized)[0]
model_rent_forecast

y.iloc[0]
```

What You Need to Do

Objective:

Improve the R^2 and MSE metric for the Rent Price Prediction Problem

Possible Things To Experiment On:

- Other **preprocessing methods**
- Conduct **feature engineering** (add, create, delete features)
- Make use of **regularization**
- And many more!

For the Write-Up:

- Recommended to have:
  - Introduction: discussion of premise and data exploration
  - Methodology: details of overall methodology
  - Experiments: explanation of various trials and experiments
  - Results and Analysis: discussion of why the results came to be with some additional analysis
  - Conclusions & Recommendations: highlight of write-up, thoughts, improvements