SEN4018 Data Science with Python Term Project

Predicting House Prices

Burak Hotan 1608186

Ekin Karadağ 1602428

Aybars Dorman 1806594

Emre Çağıltı 1401213

Saudat Alishayeva 1601389

Dataset Used

We could not pull data from websites like sahibinden.com, hurriyetemlak.com via APIs. Thus, We manually entered the data we found on sahibinden.com.

Our dataset consists total of 127 houses. The dataset has the following features; County, Price (in TL), Net Square Meter, Gross Square Meter, Room Count, Building Age, Floor Location (The floor it is located on), Total Floors (The building floor count), Balcony (Boolean), With Furniture(Boolean).

Codes Link:

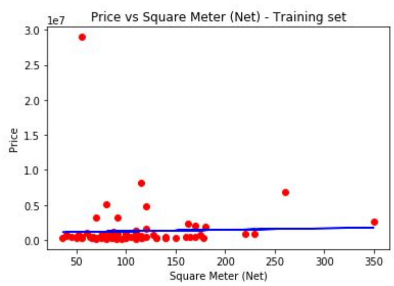
Without Polynomial:

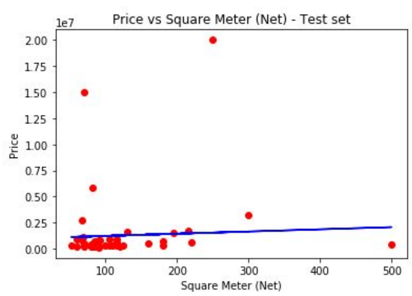
https://drive.google.com/open?id=1BKzPqHiuVSkNrzhpY9R8ynNu-mqNW-bY

Polynomial:

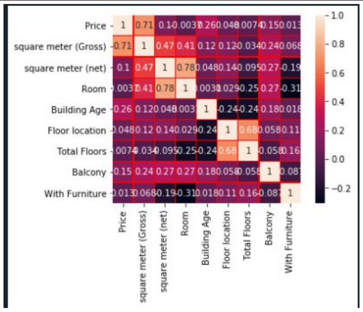
https://docs.google.com/document/d/1JQZq9NUZ3xKKCmCDW4LMkwz3Ou15idHmaXGFs8jLpFM/

Data Visualization

<table>
<tr><td>Simple Linear Regression</td><td>




</td></tr>
<tr><td>Multiple Linear Regression</td><td></td></tr>
</table>



Backward Elimination

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.024
Model:                            OLS   Adj. R-squared:                 -0.008
Method:                 Least Squares   F-statistic:                    0.7369
Date:                Wed, 27 May 2020   Prob (F-statistic):              0.569
Time:                        11:47:54   Log-Likelihood:                -2091.2
No. Observations:                 127   AIC:                             4192.
Df Residuals:                     122   BIC:                             4207.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.514e+05      8e+05      0.439      0.661   -1.23e+06    1.94e+06
x1          4.555e+04   1.04e+05      0.437      0.663   -1.61e+05    2.52e+05
x2         -1.413e+04      7e+04     -0.202      0.840   -1.53e+05    1.24e+05
x3          1.144e+06   7.24e+05      1.580      0.117    -2.9e+05    2.58e+06
x4          1.631e+05   6.33e+05      0.258      0.797   -1.09e+06    1.42e+06
==============================================================================
Omnibus:                      182.734   Durbin-Watson:                   1.400
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7969.253
Skew:                           5.716   Prob(JB):                         0.00
Kurtosis:                      40.085   Cond. No.                         37.7
==============================================================================
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1          4.145e+04   7.06e+04      0.587      0.558   -9.83e+04    1.81e+05
x2          1.368e+06   4.91e+05      2.785      0.006    3.96e+05    2.34e+06
x3          2.483e+05   5.72e+05      0.434      0.665   -8.83e+05    1.38e+06
==============================================================================
Omnibus:                      181.203   Durbin-Watson:                   1.403
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7708.377
Skew:                           5.643   Prob(JB):                         0.00
Kurtosis:                      39.460   Cond. No.                         12.1
==============================================================================
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1          5.043e+04   6.73e+04      0.749      0.455   -8.28e+04    1.84e+05
x2          1.435e+06   4.64e+05      3.089      0.002    5.15e+05    2.35e+06
==============================================================================
Omnibus:                      181.643   Durbin-Watson:                   1.415
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7846.426
Skew:                           5.659   Prob(JB):                         0.00
Kurtosis:                      39.806   Cond. No.                         9.11
==============================================================================
```

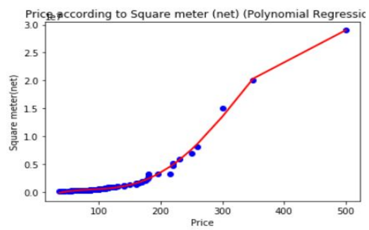
```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1          1.659e+06   3.54e+05      4.684      0.000    9.58e+05    2.36e+06
==============================================================================
```

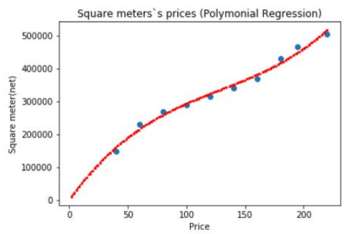
Forward Selection

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.514e+05      8e+05      0.439      0.661   -1.23e+06    1.94e+06
x1          4.555e+04   1.04e+05      0.437      0.663   -1.61e+05    2.52e+05
x2         -1.413e+04      7e+04     -0.202      0.840   -1.53e+05    1.24e+05
x3          1.144e+06   7.24e+05      1.580      0.117    -2.9e+05    2.58e+06
x4          1.631e+05   6.33e+05      0.258      0.797   -1.09e+06    1.42e+06
==============================================================================
Omnibus:                      182.734   Durbin-Watson:                   1.400
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7969.253
Skew:                           5.716   Prob(JB):                         0.00
Kurtosis:                      40.085   Cond. No.                         37.7
==============================================================================
```

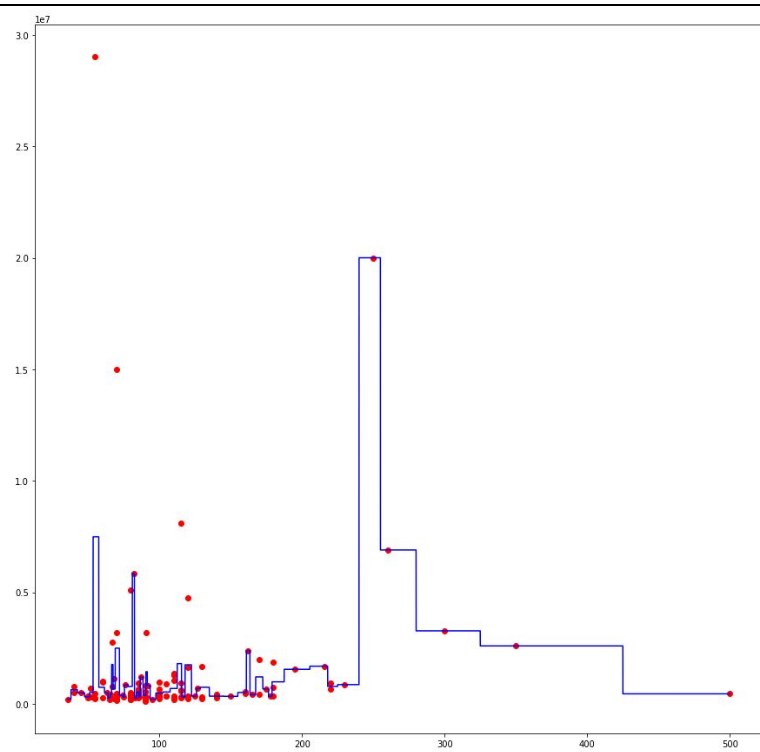
```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            1.51e+06   4.26e+05      3.544      0.001    6.67e+05    2.35e+06
x2           3.464e+05   5.45e+05      0.636      0.526   -7.32e+05    1.43e+06
==============================================================================
Omnibus:                      180.072   Durbin-Watson:                   1.425
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7446.534
Skew:                           5.597   Prob(JB):                         0.00
Kurtosis:                      38.804   Cond. No.                         1.96
==============================================================================
```

Polynomial Regression

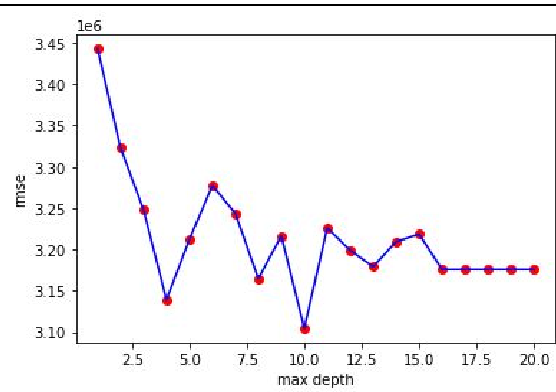




Decision Tree



Random Forest



Data Preprocessing

1. Data Formatting

Since we created our own dataset, we had the advantage of optimizing our own dataset for our use. Our dataset is read and passed to memory as a DataFrame object. Features and their type are the following:

County:	str
Price:	int
Square Meter Net:	int
Square Meter Gross:	int
Room:	int
Building Age:	int
Floor Location:	int
Total Floors:	int
Balcony:	int (either 1 or 0)
With Furniture:	int (either 1 or 0)

Feature Selection –Extraction

1. **Simple Linear Regression**

$$y = m * x + n$$

$$\Downarrow$$

$$\text{Price} = \text{Coefficient} * \text{Net Square Meter} + \text{Constant}$$

   In this algorithm, only two features are used which are Price and Net Square Meter. Price is our dependent variable and Net Square Meter is our independent variable.

2. **Multiple Linear Regression**
   Five features are used in this algorithm.Price is our dependent variable and Floor Location, Total Floors, Balcony, With Furniture are our independent variables.
3. **Polynomial Linear Regression**
   We used 1 dependent (Price) and 1 independent features (Square meter(net)).Also 2 features were represented as linear regression and Polynomial linear regression to compare x and y for finding the best way to draw a line through the data points.

4. **Decision Tree**
   In this algorithm, just two features have been selected that are Price and Net Square Meter. Price is our dependent variable and Net Square Meter is our independent variable.
5. **Random Forest**
   We used eight independent features (Net Square Meter, Gross Square Meter, Room Count, Building Age, Floor Location (The floor it is located on), Total Floors (The building floor count), Balcony (Boolean), With Furniture(Boolean)) and one dependent feature (Price)

Dataset Splitting

1. **Simple Linear Regression**
   In this algorithm, the test dataset is 1/3 of the actual dataset and the rest belongs to the training set.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0)
```

2. **Multiple Linear Regression**

   In multiple linear regression,we used the same training set and test set with simple linear regression.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0)
```

3. **Polynomial Linear Regression**

```
#p = price, s = square_meter_net
p = mydata.iloc[:, 1:2].values
s = mydata.iloc[:, 3:4].values
p = np.sort(p, axis = 0)
s = np.sort(s, axis = 0)
```

4. **Decision Tree**

```
x_dt=df.iloc[:,3:4]#Square Meter(Net)
y_dt=df.iloc[:,1:2]#Price
X_dt=x_dt.values
Y_dt=y_dt.values
```

5. **Random Forest**

```
x = ds.iloc[:, [2,3,4,5,6,7,8,9]]
y = ds.iloc[:, 1]
```

Modeling

1. **Model Training**

   a. **Simple Linear Regression**

   Code:

```
x = dataset.iloc[:, 3].values # Square meter (net)
y = dataset.iloc[:, 1].values # Price

# Create training and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0) #Splits
x_train = x_train.reshape(-1, 1) #Make 1D array
x_test = x_test.reshape(-1, 1)
y_train = y_train.reshape(-1, 1)

# Train the model using the training set
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

b. Multiple Linear Regression

```
db = pd.read_csv("SEN4018_Combined.csv")

x = db.iloc[:,[6,7,8,9]].values
y = db.iloc[:, 1].values

sns.heatmap(db.corr(),linewidth=0.2,vmax=1.0,square=True,linecolor='red',annot=True)


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0)



regressor = LinearRegression()
regressor.fit(x_train, y_train)


y_pred = regressor.predict(x_test)
```

c. Polynomial Linear Regression

```
# Fitting Polynomial Regression to the dataset

poly = PolynomialFeatures(degree = 4)

p_poly = poly.fit_transform(s)

poly.fit(p_poly, p)
lin2 = LinearRegression()
lin2.fit(p_poly, p)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

d. Decision Tree

```
r_dt=DecisionTreeRegressor(random_state=0)
r_dt.fit(X_dt,Y_dt)

X_grid = np.arange(min(X_dt), max(X_dt), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
```

e. Random forest

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 100, max_depth = 10, random_state = 0)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

2. Model Evaluation and Testing

a. Simple Linear Regression

The code creates a linear plot, according to price and net square meter of the houses. Then the price of any house can be predicted easily. The code also gives the user the ability to predict a price for the given net square meter value.

```
# Predict the test results
y_pred = regressor.predict(x_test)

# Visulize the training set results
plt.scatter(x_train, y_train, color = 'red')
plt.plot(x_train, regressor.predict(x_train), color = 'blue')
plt.title('Price vs Square Meter (Net) - Training set')
plt.xlabel('Square Meter (Net)')
plt.ylabel('Price')
plt.show()

# Visulize the test set results
plt.scatter(x_test, y_test, color = 'red')
plt.plot(x_test, regressor.predict(x_test), color = 'blue')
plt.title('Price vs Square Meter (Net) - Test set')
plt.xlabel('Square Meter (Net)')
plt.ylabel('Price')
plt.show()

coef = regressor.coef_[0][0]
intercept = regressor.intercept_[0]
print("Linear Regression Equation is:\tPrice = (", coef, " * Square Meter ) + (", intercept, ")")

userSquareMeter = int(input("Enter a square meter value to predict a price using Simple Linear Regression: "))
print("Square Meter: ", userSquareMeter, " m^2")
print("Predicted Price: ", regressor.predict([[userSquareMeter]])[0][0], " TL")
```

b. Multiple Linear Regression

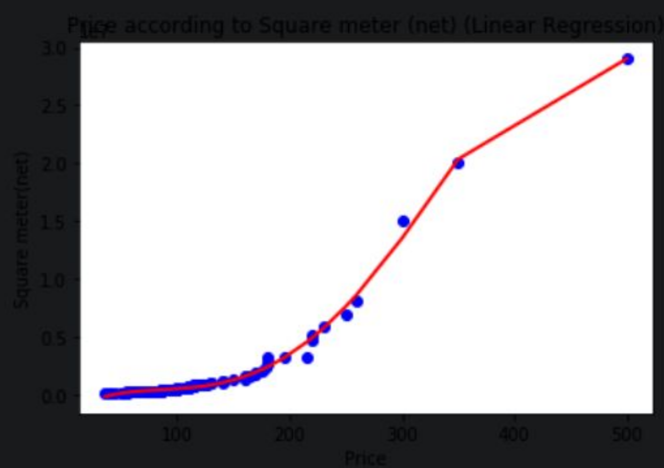
```
from sklearn import metrics

msqe = metrics.mean_squared_error(y_test, y_pred)
rmse = np.sqrt(msqe)
```

c. Polynomial Regression

```
# Visualising the Polynomial Regression results
```

```
plt.scatter(s,p, color = 'blue')
plt.plot(s, lin2.predict(poly.fit_transform(s)), color = 'red')
plt.title('Price according to Square meter (net) (Linear Regression)')
plt.xlabel('Price')
plt.ylabel('Square meter(net)')
```

```
Text(0, 0.5, 'Square meter(net)')
```




```
#Dividing the dataset into 2 components for prediction
X = mydata.iloc[:, 1:2].values # Prices
Y = mydata.iloc[:, 3:4].values # Square Meter(Net)
```

```
# Fitting Linear Regression to the dataset
lin = LinearRegression()
lin.fit(X, Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
# Fitting Polynomial Regression to the dataset
poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)
poly.fit(X_poly, Y)
lin2 = LinearRegression()
lin2.fit(X_poly, Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
# Predicting results with Linear Regression with Attributes of Price
lin.predict(X)
```

```
       [118.34054193],
```

d. Decision Tree

```
plt.figure(figsize=(10,10))
plt.scatter(X_dt,Y_dt,color='red')
plt.plot(X_grid,r_dt.predict(X_grid),color='blue')
plt.show()
dt_pred=r_dt.predict(X_dt)
```

e. Random Forest

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 100, max_depth = 10, random_state = 0)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

As can be seen from data visualization of random forest approach we identified that one of the most important hyperparameters - maximum depth should be equal to 10 so that we have the lowest RMSE value. The simplest way to find RMSE is to use the following formula:

```
msqe = sum((y_pred - y_test) * (y_pred - y_test)) / y_test.shape[0]
rmse = np.sqrt(msqe)
```

Appendix

Codes(Polynomial has been separated):

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor



df=pd.read_csv('SEN4018_Combined.csv',sep=',')

df=df.sort_values(by=['County'], ascending=True)
```

```
df=df.reset_index(drop=True)

#Linear Regression

""" Need to pass the dataset parameter which is 'SEN4018_Combined.csv'
    You can use:        dataset = pd.read_csv('SEN4018_Combined.csv')
"""
def linearRegression(dataset):
    x = dataset.iloc[:, 3].values # Square meter (net)
    y = dataset.iloc[:, 1].values # Price

    # Create training and test sets
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0)
    x_train = x_train.reshape(-1, 1)
    x_test = x_test.reshape(-1, 1)
    y_train = y_train.reshape(-1, 1)

    # Train the model using the training set
    regressor = LinearRegression()
    regressor.fit(x_train, y_train)

    # Predict the test results
    y_pred = regressor.predict(x_test)

    # Visulize the training set results
    plt.scatter(x_train, y_train, color = 'red')
    plt.plot(x_train, regressor.predict(x_train), color = 'blue')
    plt.title('Price vs Square Meter (Net) - Training set')
    plt.xlabel('Square Meter (Net)')
    plt.ylabel('Price')
    plt.show()

    # Visulize the test set results
```

```
    plt.scatter(x_test, y_test, color = 'red')
    plt.plot(x_test, regressor.predict(x_test), color = 'blue')
    plt.title('Price vs Square Meter (Net) - Test set')
    plt.xlabel('Square Meter (Net)')
    plt.ylabel('Price')
    plt.show()

    coef = regressor.coef_[0][0]
    intercept = regressor.intercept_[0]
    print("Linear Regression Equation is:\tPrice = (", coef, " * Square Meter ) + (", intercept, ")")

    userSquareMeter = int(input("Enter a square meter value to predict a price using Simple Linear Regression: "))
    print("Square Meter: ", userSquareMeter, " m^2")
    print("Predicted Price: ", regressor.predict([[userSquareMeter]])[0][0], " TL")

# Predict prices according to net areas of houses in square meter
linearRegression(df)

#Multiple Linear Regression

import seaborn as sns
import statsmodels.api as sm

db = pd.read_csv("SEN4018_Combined.csv")

x = db.iloc[:,[6,7,8,9]].values
y = db.iloc[:, 1].values

sns.heatmap(db.corr(),linewidth=0.2,vmax=1.0,square=True,linecolor='red',annot=True)


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0)
```

```
regressor = LinearRegression()
regressor.fit(x_train, y_train)


y_pred = regressor.predict(x_test)

from sklearn import metrics

msqe = metrics.mean_squared_error(y_test, y_pred)
rmse = np.sqrt(msqe)


#backward elimination
x=np.append(np.ones((127,1)).astype(int),values=x,axis=1)

x_opt=x[:,:]
r_ols=sm.OLS(endog=y,exog=x_opt)
r=r_ols.fit()
print(r.summary())

x_opt=x[:,[1,3,4]]
r_ols=sm.OLS(endog=y,exog=x_opt)
r=r_ols.fit()
print(r.summary())

x_opt=x[:,[1,3]]
r_ols=sm.OLS(endog=y,exog=x_opt)
r=r_ols.fit()
print(r.summary())
x_opt=x[:,[3]]
r_ols=sm.OLS(endog=y,exog=x_opt)
```

```
r=r_ols.fit()
print(r.summary())

#Forward Selection
x_opt=x[:,0:5]
r_ols=sm.OLS(endog=y,exog=x_opt)
r=r_ols.fit()
print(r.summary())

x_opt=x[:,3:5]
r_ols=sm.OLS(endog=y,exog=x_opt)
r=r_ols.fit()
print(r.summary())

#Decision Tree

from sklearn.tree import DecisionTreeRegressor
x_dt=df.iloc[:,3:4]#Square Meter(Net)
y_dt=df.iloc[:,1:2]#Price
X_dt=x_dt.values
Y_dt=y_dt.values

r_dt=DecisionTreeRegressor(random_state=0)
r_dt.fit(X_dt,Y_dt)

X_grid = np.arange(min(X_dt), max(X_dt), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))

plt.figure(figsize=(10,10))
plt.scatter(X_dt,Y_dt,color='red')
plt.plot(X_grid,r_dt.predict(X_grid),color='blue')
plt.show()
dt_pred=r_dt.predict(X_dt)
```

```
#Random Forest

ds = pd.read_csv('SEN4018_Combined.csv')
print(ds.isnull().sum())

x = ds.iloc[:, [2,3,4,5,6,7,8,9]]
y = ds.iloc[:, 1]

#max depth versus error
md = 20;
md_errors = np.zeros(md)

#split dataset into train and test splits

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 1/3, random_state = 0)

for i in range(1, md+1):
    regressor = RandomForestRegressor(n_estimators = 100, max_depth = i, random_state = 0)
    regressor.fit(x_train, y_train)
    y_pred = regressor.predict(x_test)
    #finding error
    msqe = sum((y_pred - y_test) * (y_pred - y_test)) / y_test.shape[0]
    md_errors[i-1] = np.sqrt(msqe)


plt.scatter(range(1, md+1), md_errors, color = 'red')
plt.plot(range(1, md+1), md_errors, color = 'blue')
plt.xlabel('max depth')
plt.ylabel('rmse')
plt.show()
```

```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 100, max_depth = 10, max_features = 0.5, min_samples_split = 5,
random_state = 0)

regressor.fit(x_train, y_train)

y_pred = regressor.predict(x_test)

msqe = sum((y_pred - y_test) * (y_pred - y_test)) / y_test.shape[0]

rmse = np.sqrt(msqe)
```

Polynomial Code:

```
# In[258]:


# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.preprocessing import PolynomialFeatures


# In[259]:


#Advantages of using Polynomial Regression:

   #Broad range of function can be fit under it.

   #Polynomial basically fits wide range of curvature.

   #Polynomial provides the best approximation of the relationship between dependent and independent variable.


# In[260]:


#Disadvantages of using Polynomial Regression

   #These are too sensitive to the outliers.

   #The presence of one or two outliers in the data can seriously affect the results of a nonlinear analysis.
```

#In addition there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.

```
# In[261]:


# Importing the dataset
mydata = pd.read_csv('SEN4018_Combined.csv', sep = ',')



# In[262]:


mydata


# In[263]:


df = mydata.iloc[:, 1:]


# In[264]:


mydata = mydata.sort_values(by=['County'], ascending=True)


# In[265]:


mydata.info()


# In[266]:


#p = price, s = square_meter_net
p = mydata.iloc[:, 1:2].values
s = mydata.iloc[:, 3:4].values
p = np.sort(p, axis = 0)
s = np.sort(s, axis = 0)
```

```
# In[267]:


# Fitting Linear Regression to the dataset


# In[268]:


lin = LinearRegression()


# In[269]:


lin.fit(s, p)


# In[270]:


# Fitting Polynomial Regression to the dataset


# In[271]:


poly = PolynomialFeatures(degree = 4)


# In[272]:


p_poly = poly.fit_transform(s)


# In[273]:


poly.fit(p_poly, p)
lin2 = LinearRegression()
lin2.fit(p_poly, p)


# In[274]:


# Visualising the Linear Regression results
```

```
# In[275]:


plt.scatter(s, p, color = 'blue')
plt.plot(s, lin.predict(s), color = 'red')
plt.title('Price according to Square meter (net) (Linear Regression)')
plt.ylabel('Price')
plt.xlabel('square meter (net)')
plt.rc('lines', linewidth=2)
plt.show()


# In[276]:


# Visualising the Polynomial Regression results


# In[277]:


plt.scatter(s,p, color = 'blue')
plt.plot(s, lin2.predict(poly.fit_transform(s)), color = 'red')
plt.title('Price according to Square meter (net) (Polynomial  Regression)')
plt.xlabel('Price')
plt.ylabel('Square meter(net)')


# In[278]:


#Dividing the dataset into 2 components for prediction
X = mydata.iloc[:, 1:2].values # Prices
Y = mydata.iloc[:, 3:4].values # Square Meter(Net)


# In[279]:


# Fitting Linear Regression to the dataset
lin = LinearRegression()
```

```
lin.fit(X, Y)


# In[280]:


# Fitting Polynomial Regression to the dataset
poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)
poly.fit(X_poly, Y)
lin2 = LinearRegression()
lin2.fit(X_poly, Y)


# In[281]:


# Predicting results with Linear Regression with Attributes of Price
lin.predict(X)



# In[282]:


# Predicting results with Linear Regression with Attributes of Square meter(net)
lin.predict(Y)


# In[283]:


# Predicting a new result with Polynomial Regression
lin2.predict(poly.fit_transform(X,Y))


# In[286]:


#10 Values for visualization in terms of polynomial regression
p = [148000,230000,268000,290000,315000,340000,369000,430000,465000,505000]
s = [40,60,80,100,120,140,160,180,195,220]
mymodel = numpy.poly1d(numpy.polyfit(s, p, 3))
```

```
myline = numpy.linspace(2,220,100)

plt.scatter(s,p)

plt.plot(myline, mymodel(myline),linewidth=2,linestyle="--",marker="o",markersize=2,color="r")

plt.title('Square meters`s prices (Polymonial Regression)')

plt.xlabel('Price')

plt.ylabel('Square meter(net)')

plt.show()
```