HOUSE PRICE PREDICTION

Machine Learning Project - Phase 2

Submitted by:
Vishvesh Rao
(AM.EN.U4CSE19161)
S5 CSE B

ML PROBLEM DESCRIPTION

Formal description

- Task (T): Predict the price of house
- Experience(E): A dataset which has the prices of houses from various in a particular area
- Performance(P): The number of house prices predicted correctly out of all the prices given

Assumptions

- This assumes that external factors such as weather, crime rates etc do not affect housing prices
- This model also does not take into account the location of the houses such as distance to the nearest form of public transportation or any other type of establishments.
- The data provided to us is from 1990 ranging to 2019 so we would be making prediction using the trends found in that dataset

DATASET

This Dataset is the Boston Housing Prices, Boston is a suburb in USA.

The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1990. The data pertains to the houses found in a given California district and some summary stats about them based on the 1990 census data. It consistis of 20640 instances

The attributes are defined as follows:

1. **longitude**: A measure of how far west a house is; a higher value is farther west
2. **latitude**: A measure of how far north a house is; a higher value is farther north
3. **housingMedianAge**: Median age of a house within a block; a lower number is a newer building
4. **totalRooms**: Total number of rooms within a block
5. **totalBedrooms**: Total number of bedrooms within a block
6. **population**: Total number of people residing within a block
7. **households**: Total number of households, a group of people residing within a home unit, for a block
8. **medianIncome**: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
9. **medianHouseValue**: Median house value for households within a block (measured in US Dollars)
10. **oceanProximity**: Location of the house w.r.t ocean/sea

PRE-PROCESSING

Initial Dataset had 8 features (excluding target feature) and consisted of 20,640 entries/samples.

Pre-processing steps

1. *Feature analysis*

checked all the features for null value and number of distinct values present in a feature based on these criterias removed a feature till there were no null values and all features contained a good amount of distinct values

Also checked the correlations between the features and plotted a corelation matrix.

2. *Feature Modifications*

Certain features were not useful/useable in their present state. The “Longitude” *and* “Latitude” *attributes were not easily understandable so those were used to get exact* “Road” *and* “county” *names of each data sample and these and these two attributes were added to the dataframe after removing* “Longitude” *and* “Latitude”.

3. *Filing Missing Values*

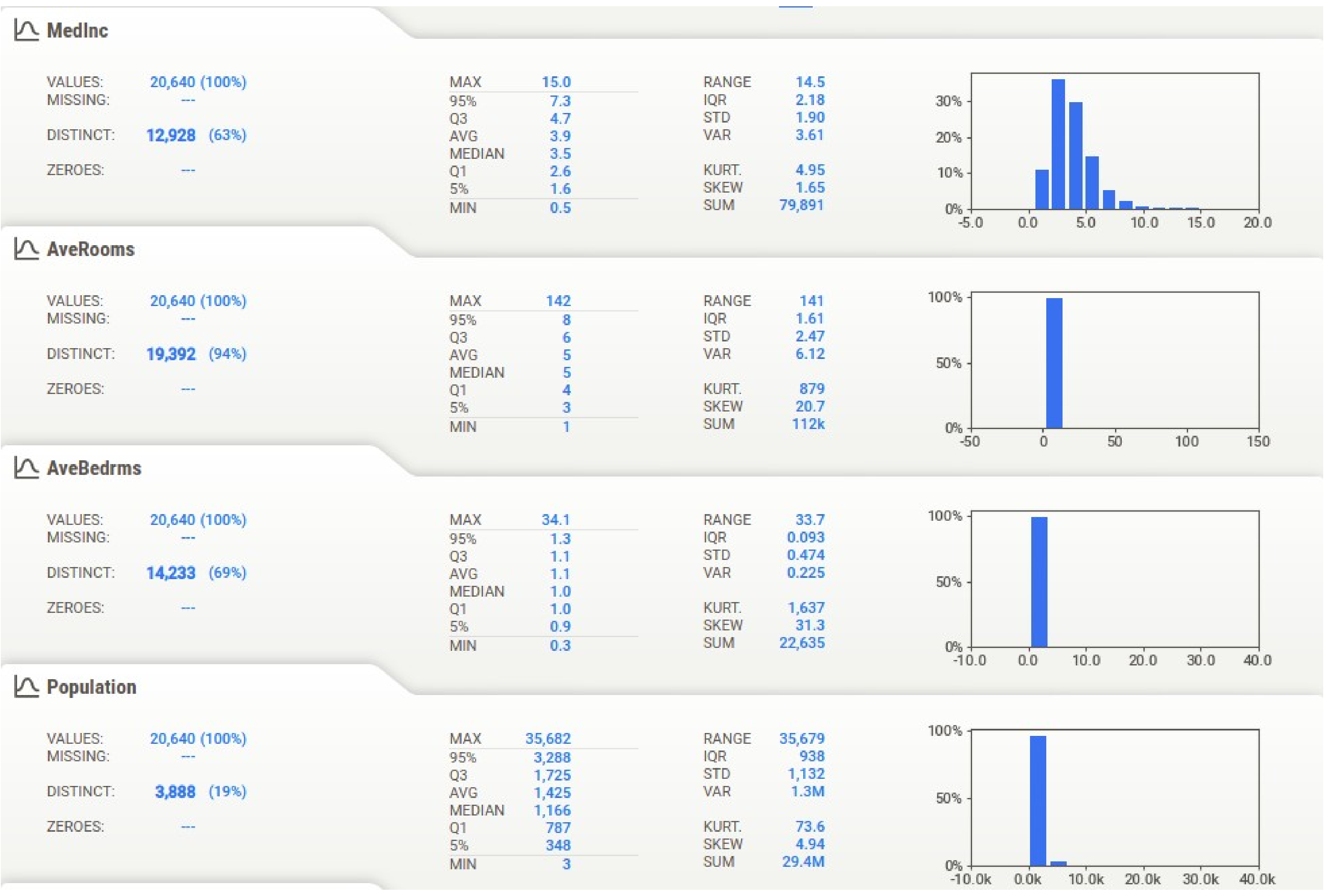
The “road” and “county” attributes had numerous missing values after geoplotting of the co-ordinates was done. Consequently Logistic Regression specifically SGDC (Stochastic Gradient Descent Classifier) was used to classify the missing road and county values.

All above processes are explained in detail with pictorial/graphical representaion of data in Phase-1 Document

DATA SUMMARIZATION AND VISUALIZATION

Here we will look at each of the individual attributes in detail analysing various statistical parameters and bar graphs.

This is the cleaned/processed data which is now ready to be split into training and testing data and therafter to be fed into the ML algorithms



MedInc

VALUES:	20,640 (100%)	MAX	15.0	RANGE	14.5
MISSING:	---	95%	7.3	IQR	2.18
DISTINCT:	12,928 (63%)	Q3	4.7	STD	1.90
ZEROES:	---	AVG	3.9	VAR	3.61
		MEDIAN	3.5	KURT.	4.95
		Q1	2.6	SKEW	1.65
		5%	1.6	SUM	79,891
		MIN	0.5		

AveRooms

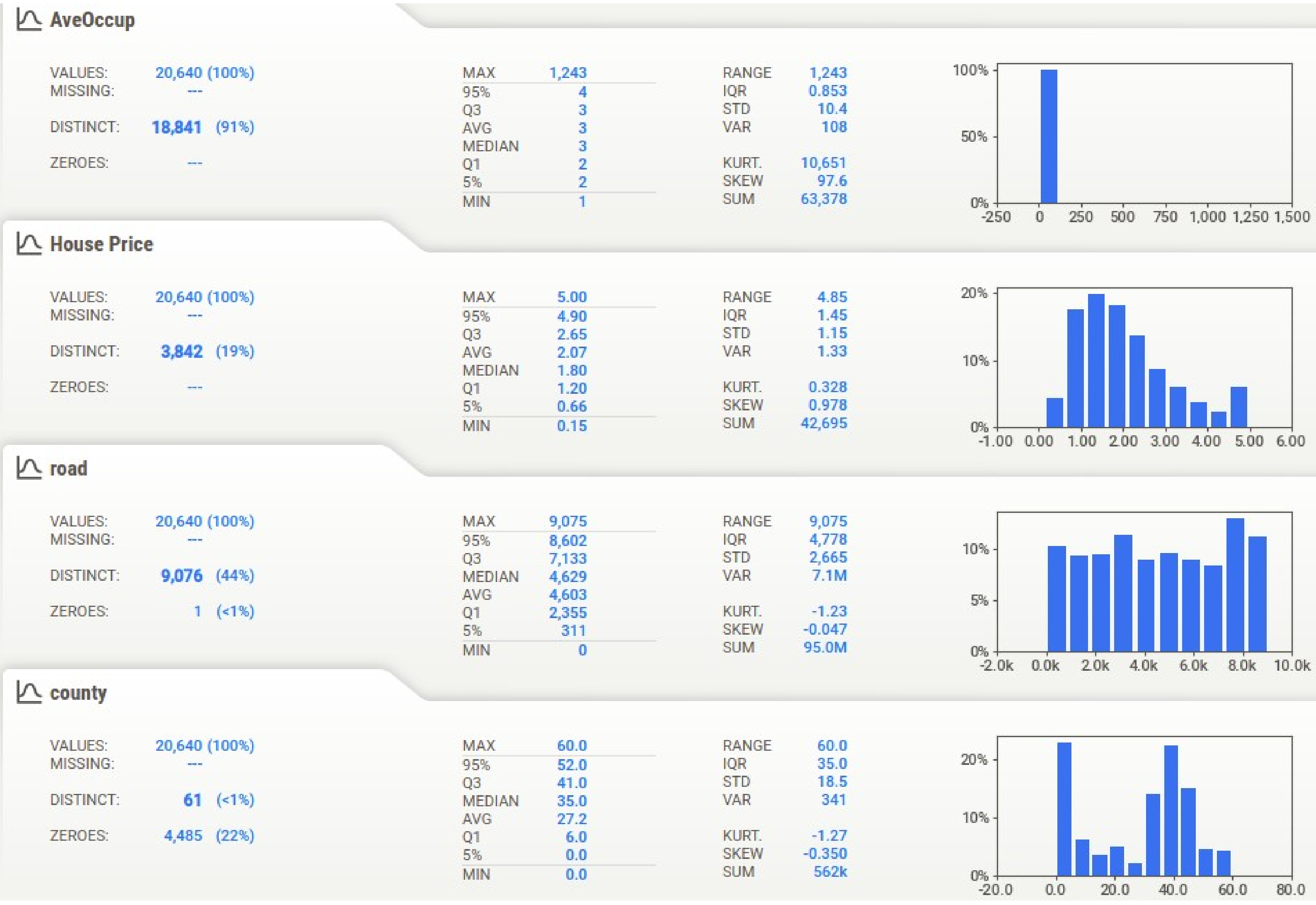
VALUES:	20,640 (100%)	MAX	142	RANGE	141
MISSING:	---	95%	8	IQR	1.61
DISTINCT:	19,392 (94%)	Q3	6	STD	2.47
ZEROES:	---	AVG	5	VAR	6.12
		MEDIAN	5	KURT.	879
		Q1	4	SKEW	20.7
		5%	3	SUM	112k
		MIN	1		

AveBedrms

VALUES:	20,640 (100%)	MAX	34.1	RANGE	33.7
MISSING:	---	95%	1.3	IQR	0.093
DISTINCT:	14,233 (69%)	Q3	1.1	STD	0.474
ZEROES:	---	AVG	1.1	VAR	0.225
		MEDIAN	1.0	KURT.	1,637
		Q1	1.0	SKEW	31.3
		5%	0.9	SUM	22,635
		MIN	0.3		

Population

VALUES:	20,640 (100%)	MAX	35,682	RANGE	35,679
MISSING:	---	95%	3,288	IQR	938
DISTINCT:	3,888 (19%)	Q3	1,725	STD	1,132
ZEROES:	---	AVG	1,425	VAR	1.3M
		MEDIAN	1,166	KURT.	73.6
		Q1	787	SKEW	4.94
		5%	348	SUM	29.4M
		MIN	3		



```
AveOccup

VALUES:    20,640 (100%)     MAX      1,243     RANGE   1,243
MISSING:   ---               95%          4     IQR     0.853
                             Q3           3     STD      10.4
DISTINCT:  18,841  (91%)     AVG          3     VAR       108
                             MEDIAN       3
ZEROES:    ---               Q1           2     KURT.  10,651
                             5%           2     SKEW     97.6
                             MIN          1     SUM    63,378
```

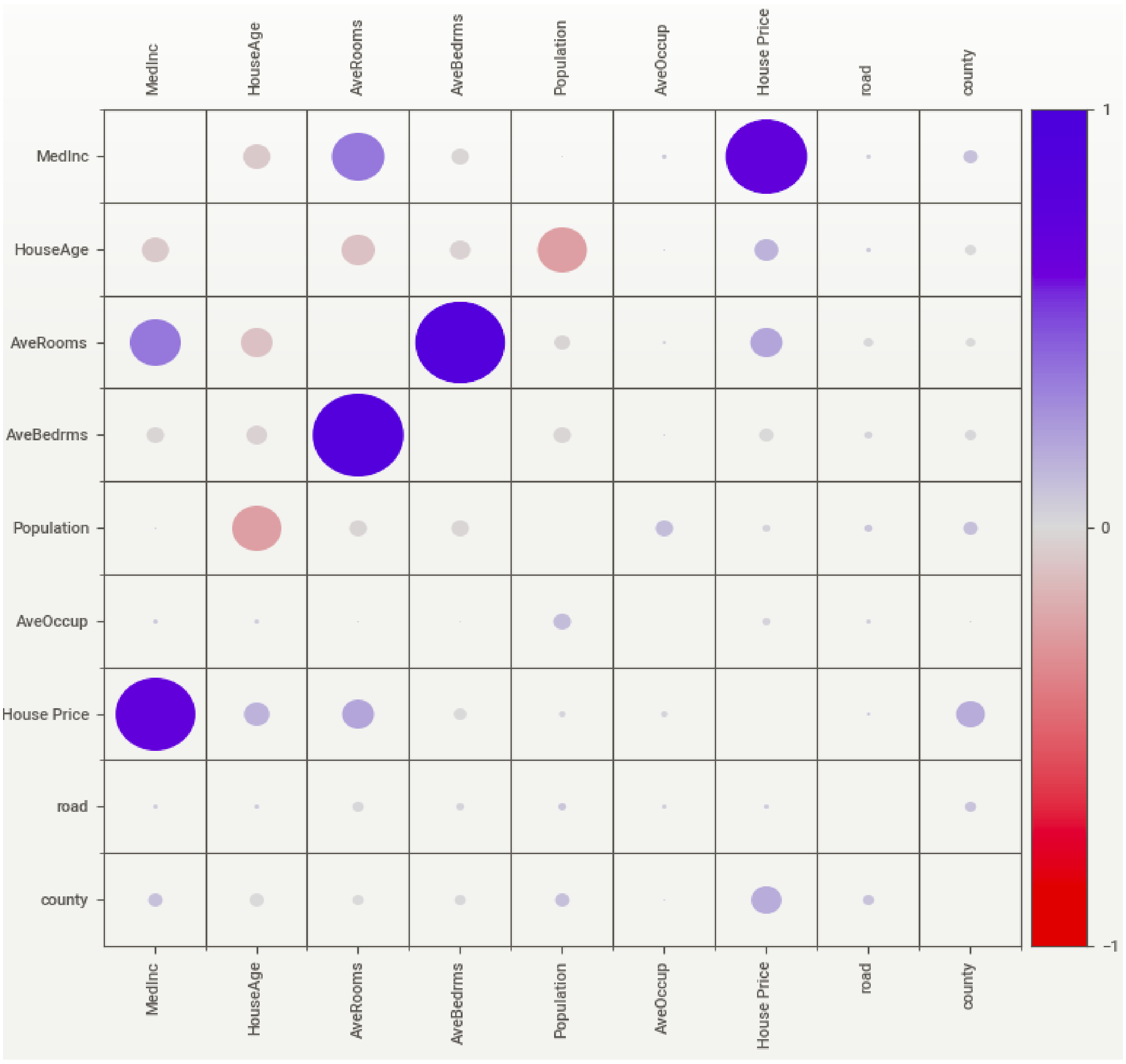
```
House Price

VALUES:    20,640 (100%)     MAX       5.00     RANGE    4.85
MISSING:   ---               95%       4.90     IQR      1.45
                             Q3        2.65     STD      1.15
DISTINCT:  3,842  (19%)      AVG       2.07     VAR      1.33
                             MEDIAN    1.80
ZEROES:    ---               Q1        1.20     KURT.   0.328
                             5%        0.66     SKEW    0.978
                             MIN       0.15     SUM    42,695
```

```
road

VALUES:    20,640 (100%)     MAX      9,075     RANGE   9,075
MISSING:   ---               95%      8,602     IQR     4,778
                             Q3       7,133     STD     2,665
DISTINCT:  9,076  (44%)      MEDIAN   4,629     VAR      7.1M
                             AVG      4,603
ZEROES:    1      (<1%)      Q1       2,355     KURT.   -1.23
                             5%         311     SKEW   -0.047
                             MIN          0     SUM     95.0M
```

```
county

VALUES:    20,640 (100%)     MAX       60.0     RANGE    60.0
MISSING:   ---               95%       52.0     IQR      35.0
                             Q3        41.0     STD      18.5
DISTINCT:  61     (<1%)      MEDIAN    35.0     VAR       341
                             AVG       27.2
ZEROES:    4,485  (22%)      Q1         6.0     KURT.   -1.27
                             5%         0.0     SKEW   -0.350
                             MIN        0.0     SUM      562k
```

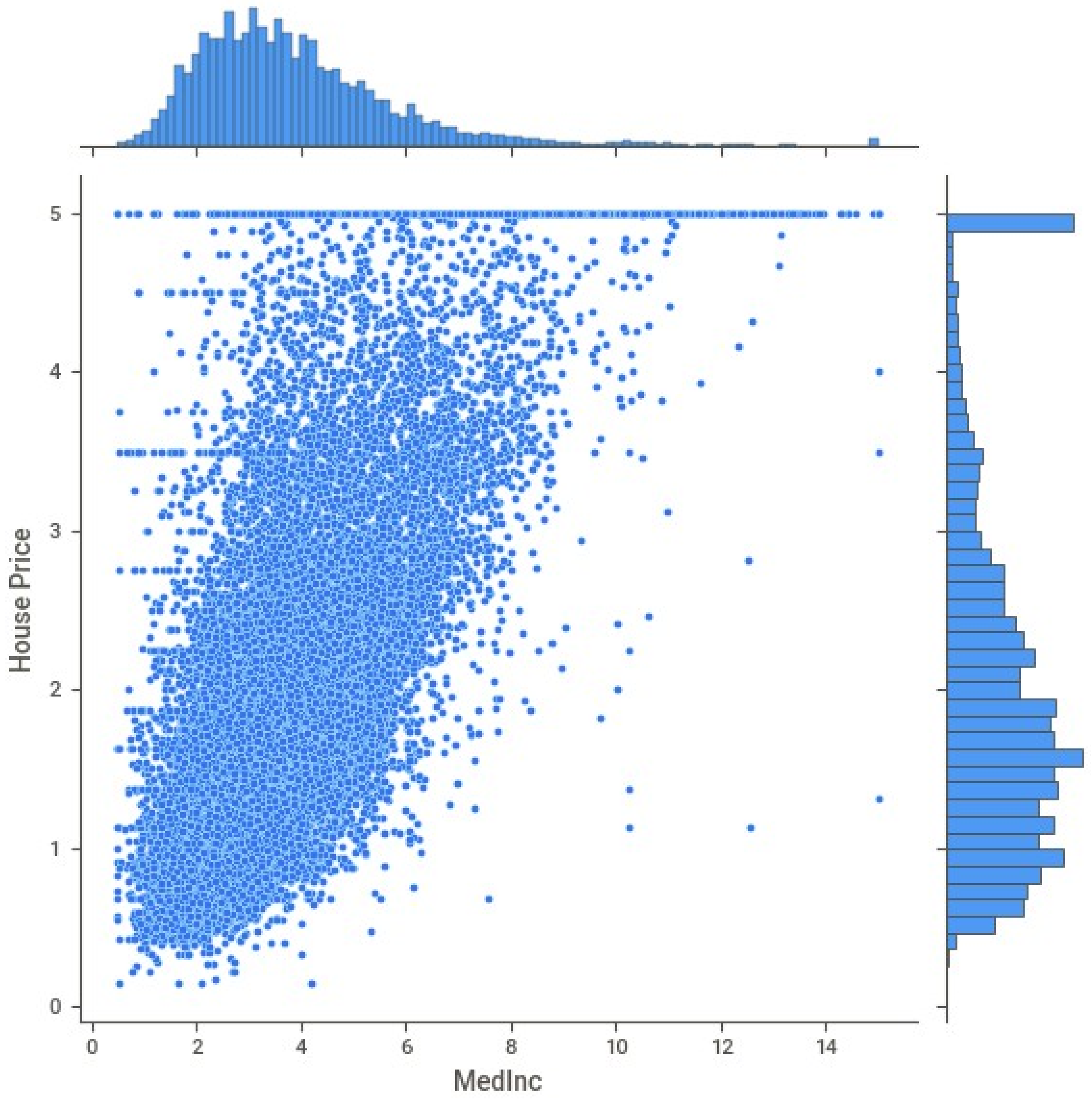
The above data gives us the max,min,null,distinct values present in each of the numerical attributes and more information like number of missing , Distinct, null values and their total sum etc.

Correlation Graph showing associations between various attributes



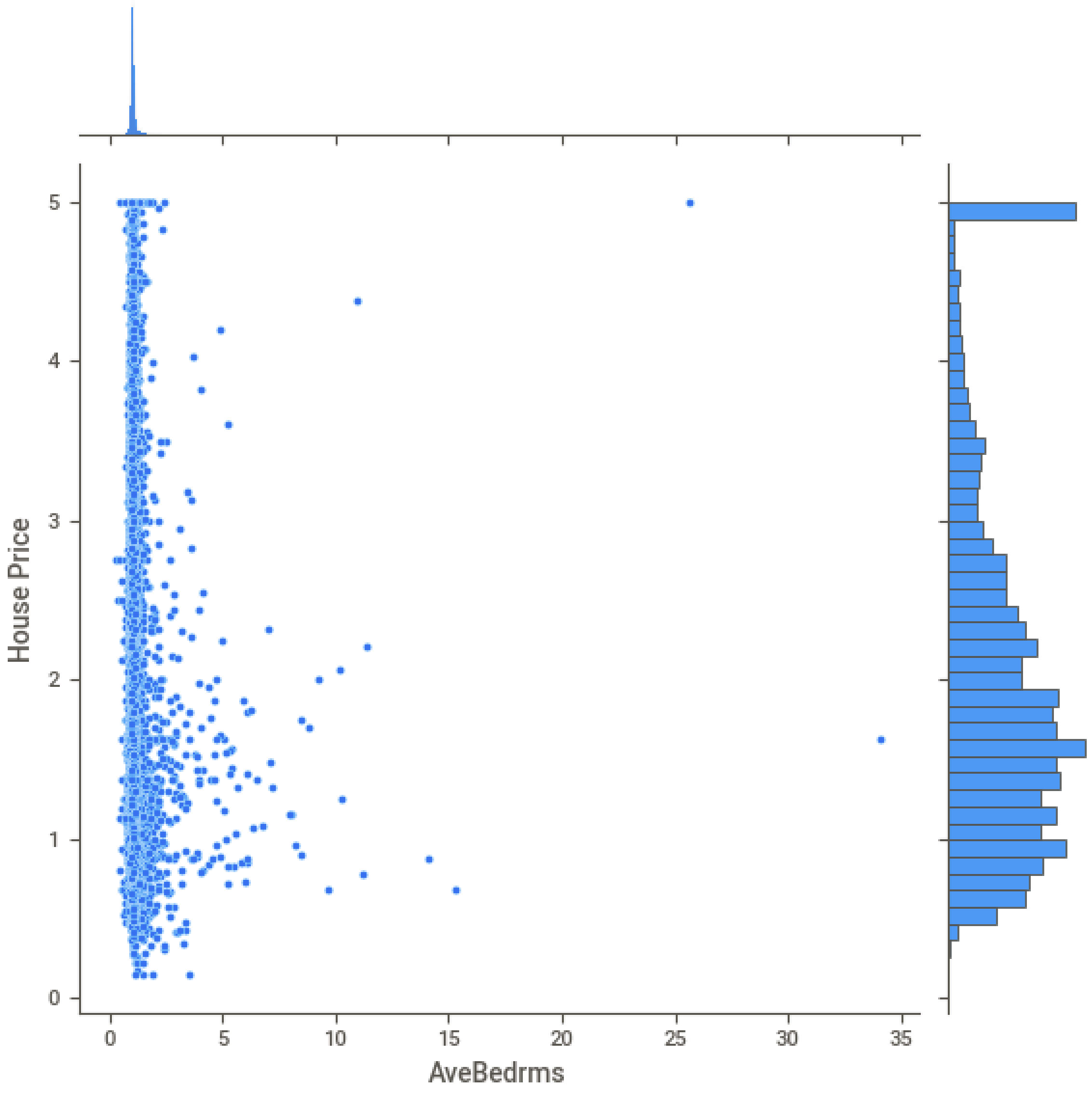
Circles *are the symmetrical numerical correlations from -1 to 1.*

Scatter plot showing the highest postive relationship out of all variables wrt to the target variable
i.e “*MedInc*” and “*House Price*”



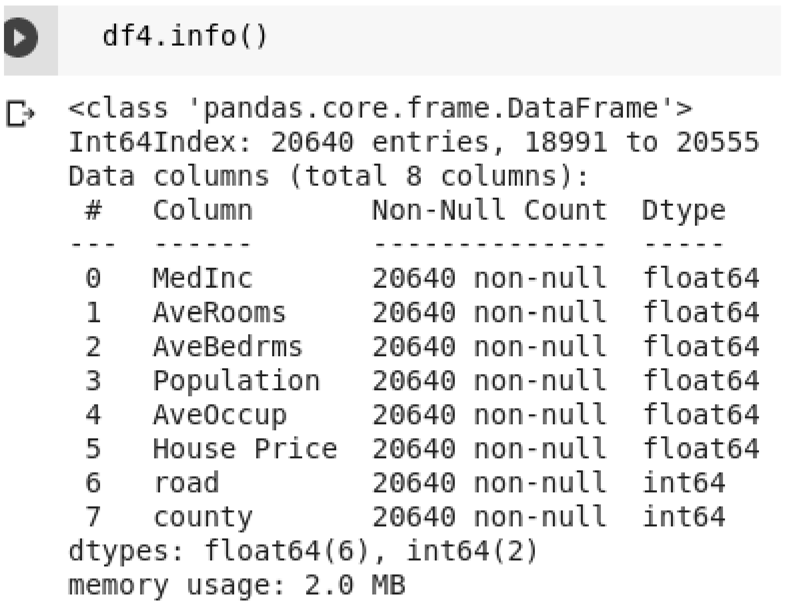
Scatter plot of target variable and “MedInc*” attribute along with their respective histograms*

Scatter plot showing the least postive relationship out of all variables wrt to the target variable
i.e “*AveBedrms*” and “*House Price*”



*Scatter plot of target variable and “*AveBedrms*” attribute along with their respective histograms*

FINAL DATASET

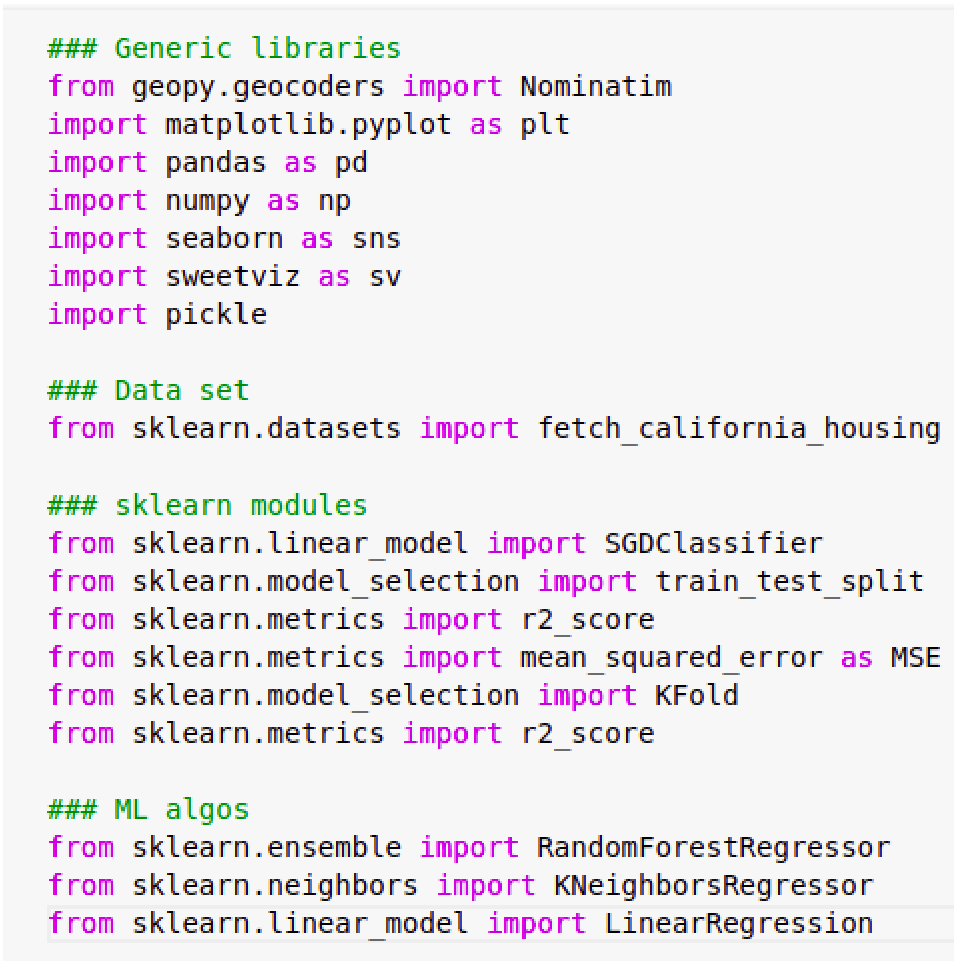


```
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20640 entries, 18991 to 20555
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   MedInc       20640 non-null  float64
 1   AveRooms     20640 non-null  float64
 2   AveBedrms    20640 non-null  float64
 3   Population   20640 non-null  float64
 4   AveOccup     20640 non-null  float64
 5   House Price  20640 non-null  float64
 6   road         20640 non-null  int64
 7   county       20640 non-null  int64
dtypes: float64(6), int64(2)
memory usage: 2.0 MB
```

With this the dataset is ready to be split into test and validate sets and therafter to be evaluated by the ML algorithm for model generation.

PYTHON PACKAGES IMPORTED



```
### Generic libraries
from geopy.geocoders import Nominatim
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import sweetviz as sv
import pickle

### Data set
from sklearn.datasets import fetch_california_housing

### sklearn modules
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import KFold
from sklearn.metrics import r2_score

### ML algos
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
```

Sweetviz and **Seaborn** were used for data analytics and plotting the graphs along with **matploitlib**

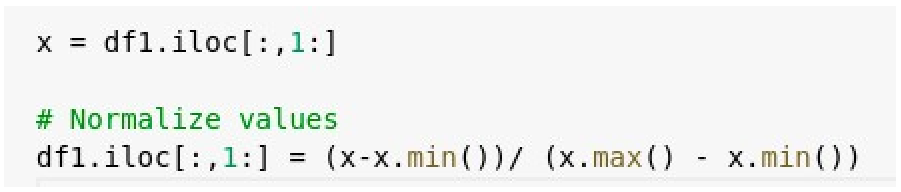
geopy was used to get the address i.e the “road” and “county” attributes from the “Latitude” and “Longitude” values.

Sklearn was used for importing the necessary dataset as well as the ml aglorithms

PRE PROCESSING

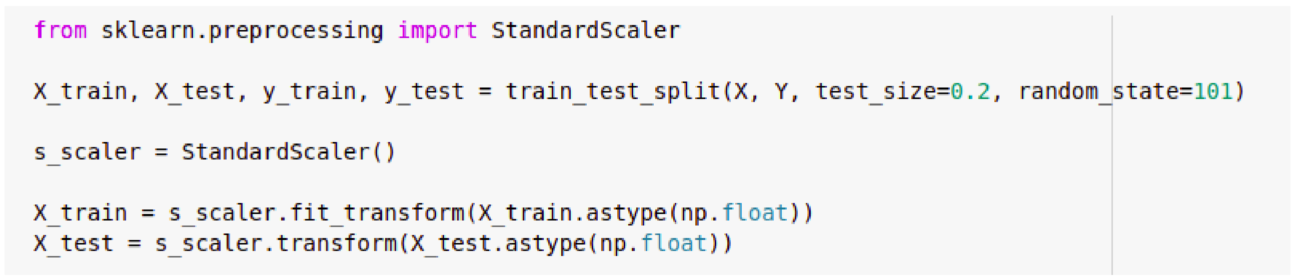
Step 1 - Normalization

Here the features are rescaled to hard and fast range of [0,1] by subtracting minimum value of the feature then dividing by range.



```
x = df1.iloc[:,1:]

# Normalize values
df1.iloc[:,1:] = (x-x.min())/ (x.max() - x.min())
```

Step 2 - Splitting the dataset and standardization



```
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=101)

s_scaler = StandardScaler()

X_train = s_scaler.fit_transform(X_train.astype(np.float))
X_test = s_scaler.transform(X_test.astype(np.float))
```

The dataset is split into training and test datasets repectively in a 80:20 split with 80% training data and 20% test

Step 3 - Choosing the Model

Three models were chosen and all of them were evaluated for determining the best one

- K Neighbour Regression (*implemented from scratch*)
- Random Forest
- Linear Regression

<u>ABOUT THE ALGOS</u>

Random Forest:- utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging.

Establishes the outcome based on the predictions of the decison trees. It predicts by tkaing the the aeverage or the mean of the output from various trees.

Linear Regression:- models the relationship between two variables by fitting a linear equation to the given data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable

A linear regression line has an equation of the form ***Y* = *a* + *bX***, where **X** is the explanatory variable and ***Y*** is the dependent variable.

The slope of the line is ***b***, and ***a*** is the intercept (the value of ***y*** when ***x*** = 0).

K Neighbour Regression :- This is a Regression algo based on k-nearest neighbors.

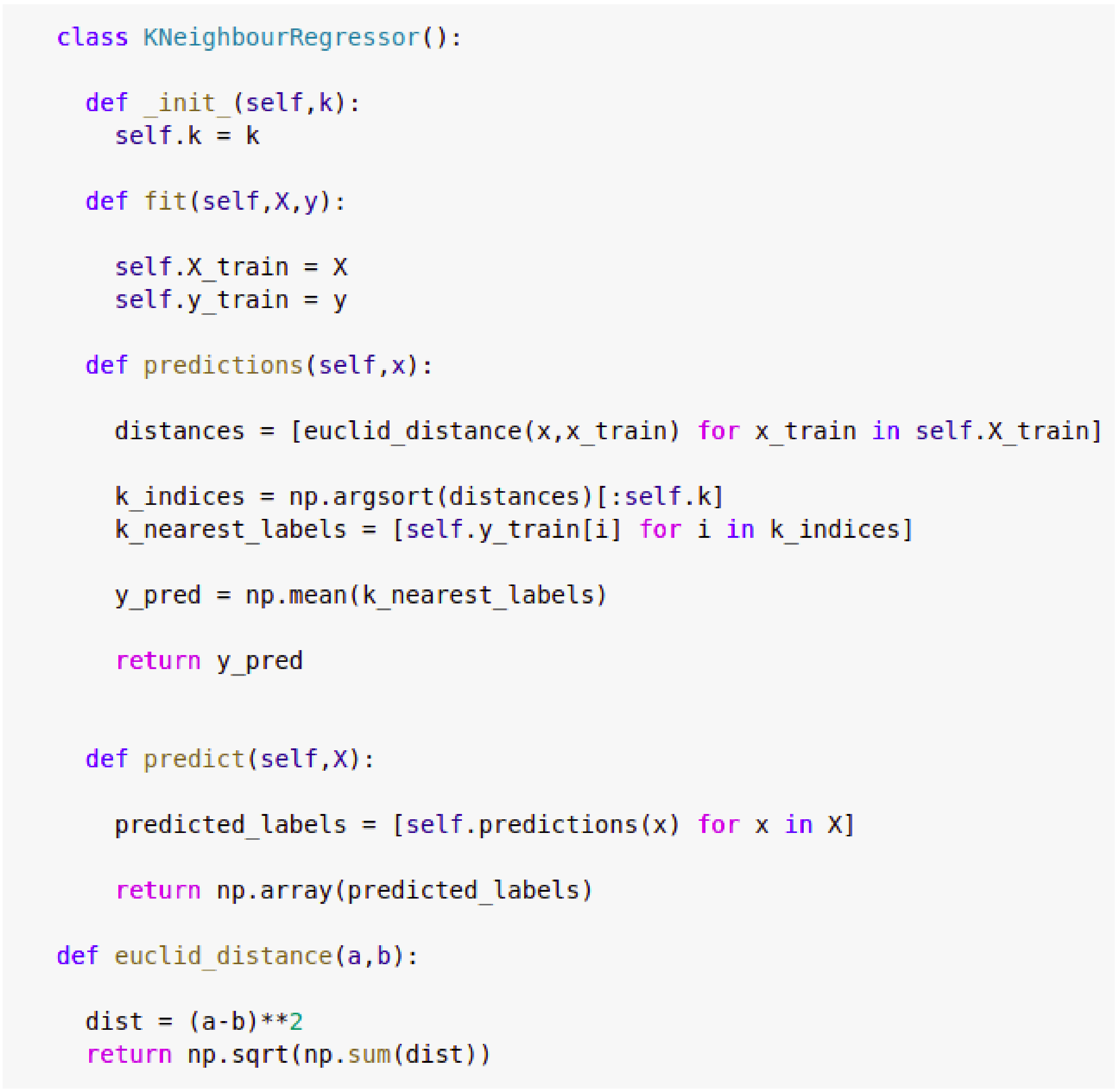
It has no model other than storing the entire dataset, so there is no learning required.

Predictions are made by searching through the entire training set for the K most similar neighbors and summarizing the output variable for those K instances. In case of regressin the it is usually the mean output variable

To measure the level of similarity a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance.

This algorithm has been implemented from scratch and the code for same is below

KNR IMPLEMENTATION

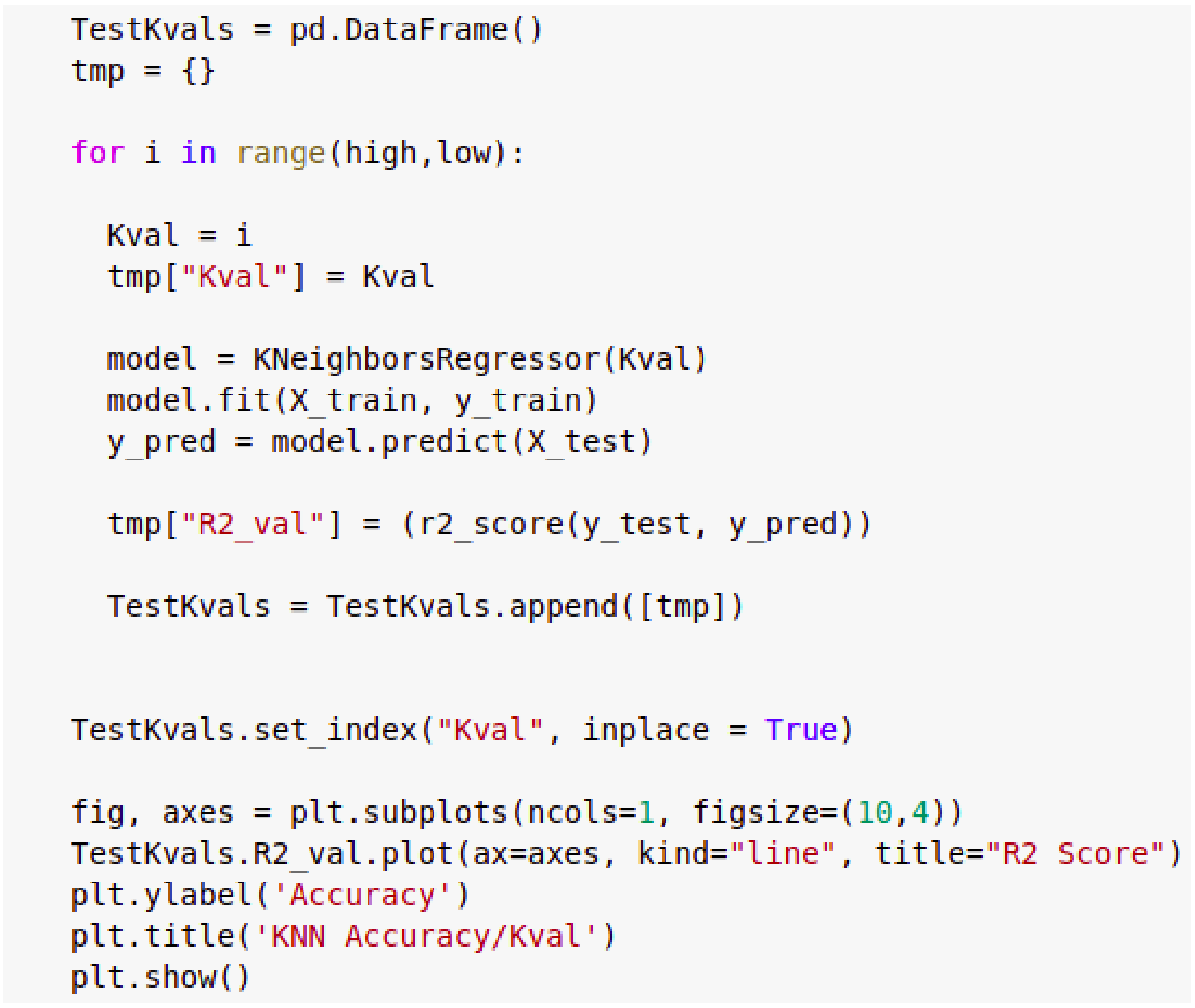


```python
class KNeighbourRegressor():

  def _init_(self,k):
    self.k = k

  def fit(self,X,y):

    self.X_train = X
    self.y_train = y

  def predictions(self,x):

    distances = [euclid_distance(x,x_train) for x_train in self.X_train]

    k_indices = np.argsort(distances)[:self.k]
    k_nearest_labels = [self.y_train[i] for i in k_indices]

    y_pred = np.mean(k_nearest_labels)

    return y_pred


  def predict(self,X):

    predicted_labels = [self.predictions(x) for x in X]

    return np.array(predicted_labels)

def euclid_distance(a,b):

  dist = (a-b)**2
  return np.sqrt(np.sum(dist))
```

MODEL EVALUATION

Choosing the best K value

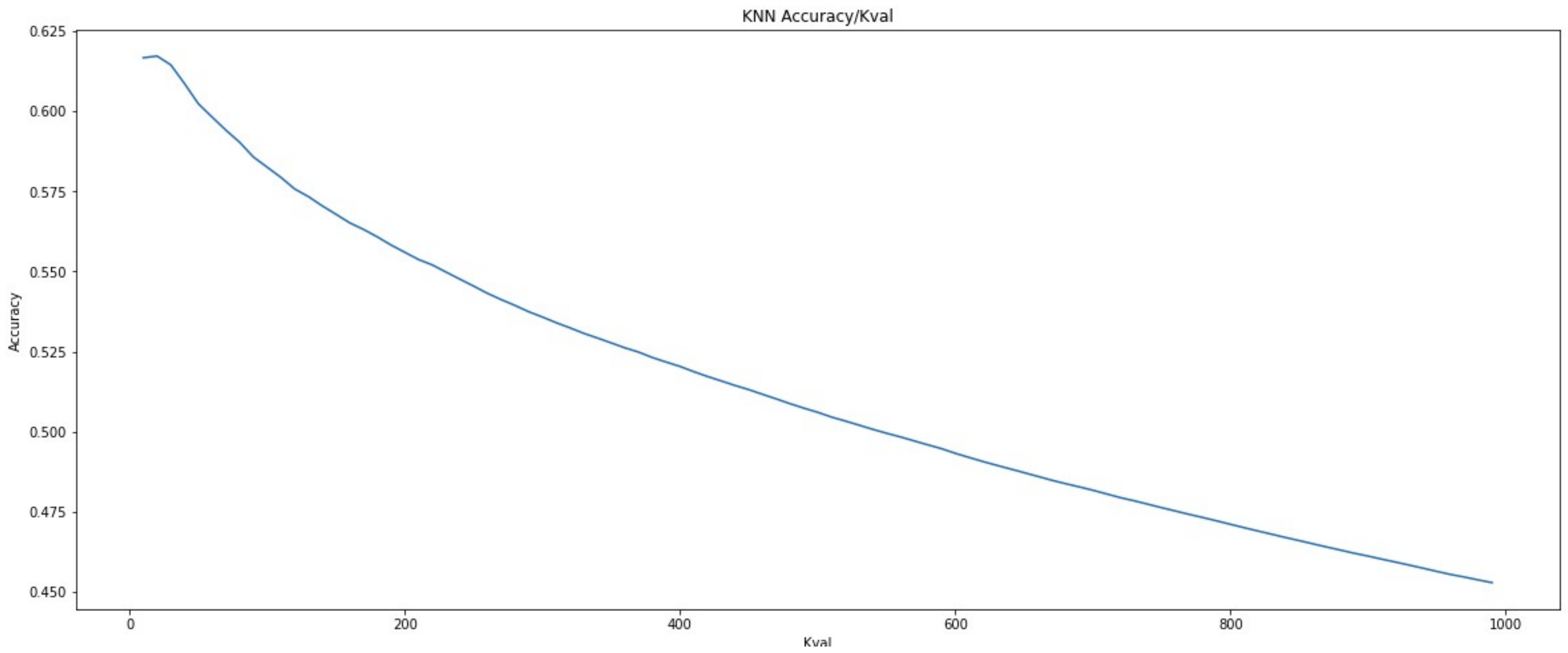
The following code and the graphs demonstrated how the correct value of K was achieved



```
TestKvals = pd.DataFrame()
tmp = {}

for i in range(high,low):

  Kval = i
  tmp["Kval"] = Kval

  model = KNeighborsRegressor(Kval)
  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)

  tmp["R2_val"] = (r2_score(y_test, y_pred))

  TestKvals = TestKvals.append([tmp])


TestKvals.set_index("Kval", inplace = True)

fig, axes = plt.subplots(ncols=1, figsize=(10,4))
TestKvals.R2_val.plot(ax=axes, kind="line", title="R2 Score")
plt.ylabel('Accuracy')
plt.title('KNN Accuracy/Kval')
plt.show()
```

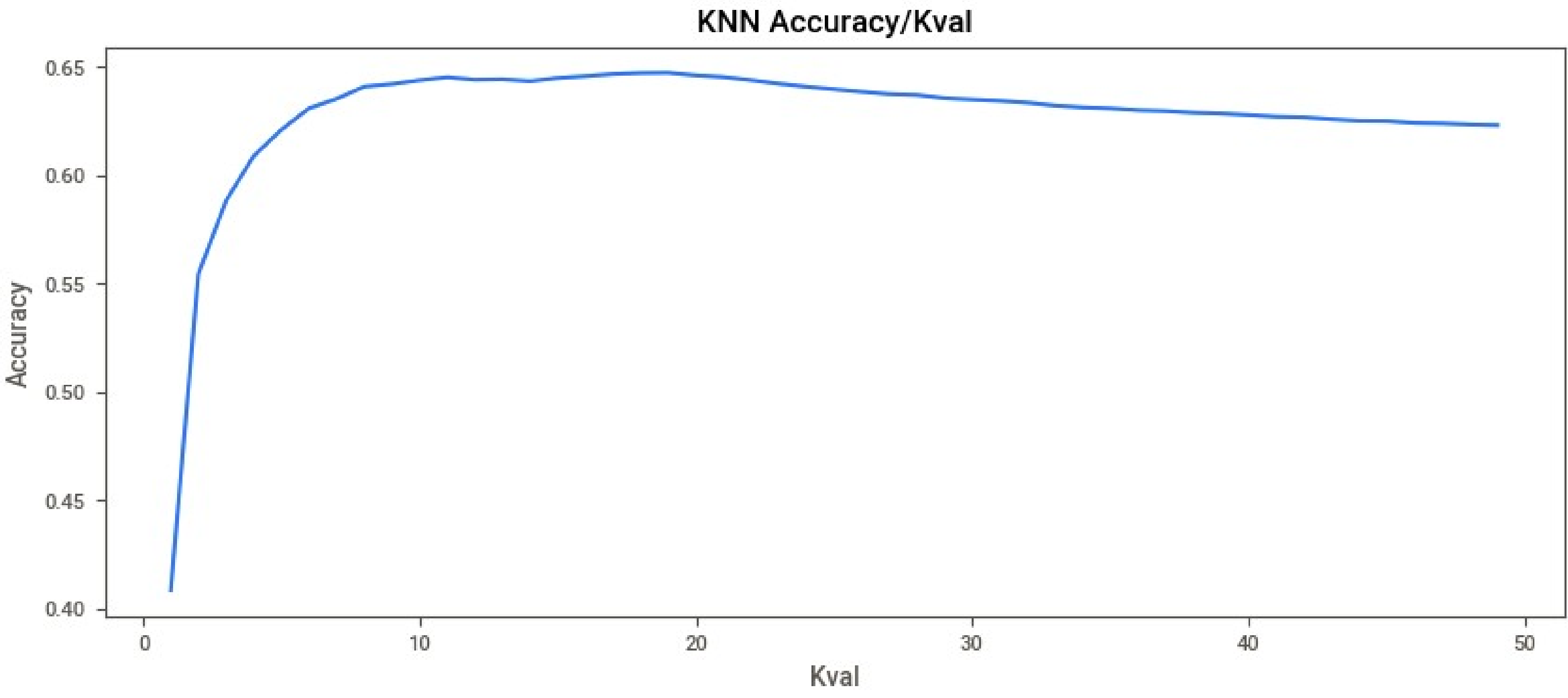
Plotting accuracy vs K value for different values of K

K from 0 to 1000



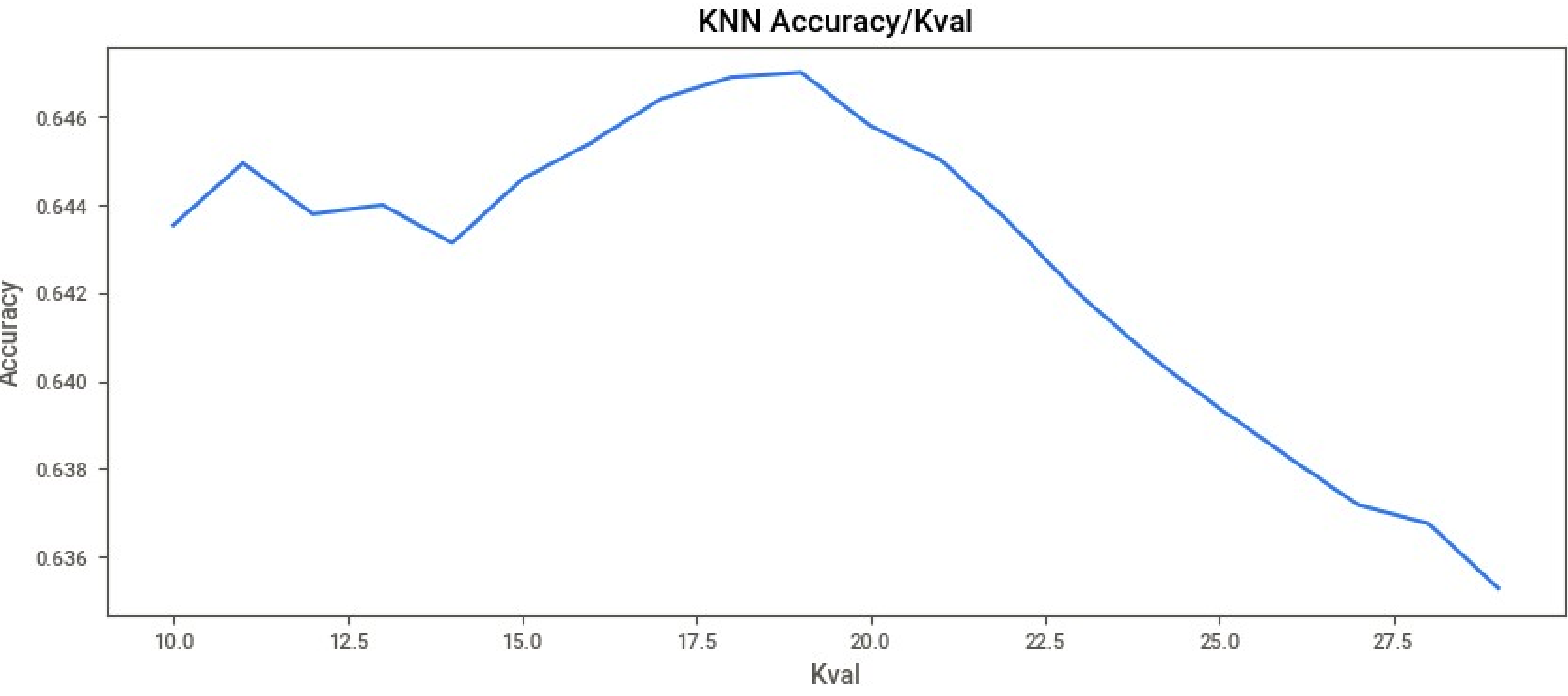
As we see here the best K value somewhere between 0 and 50 approximately

K from 0 to 50



We see the required K value is in the range of 10 to 30 approximately with value being very close to 20

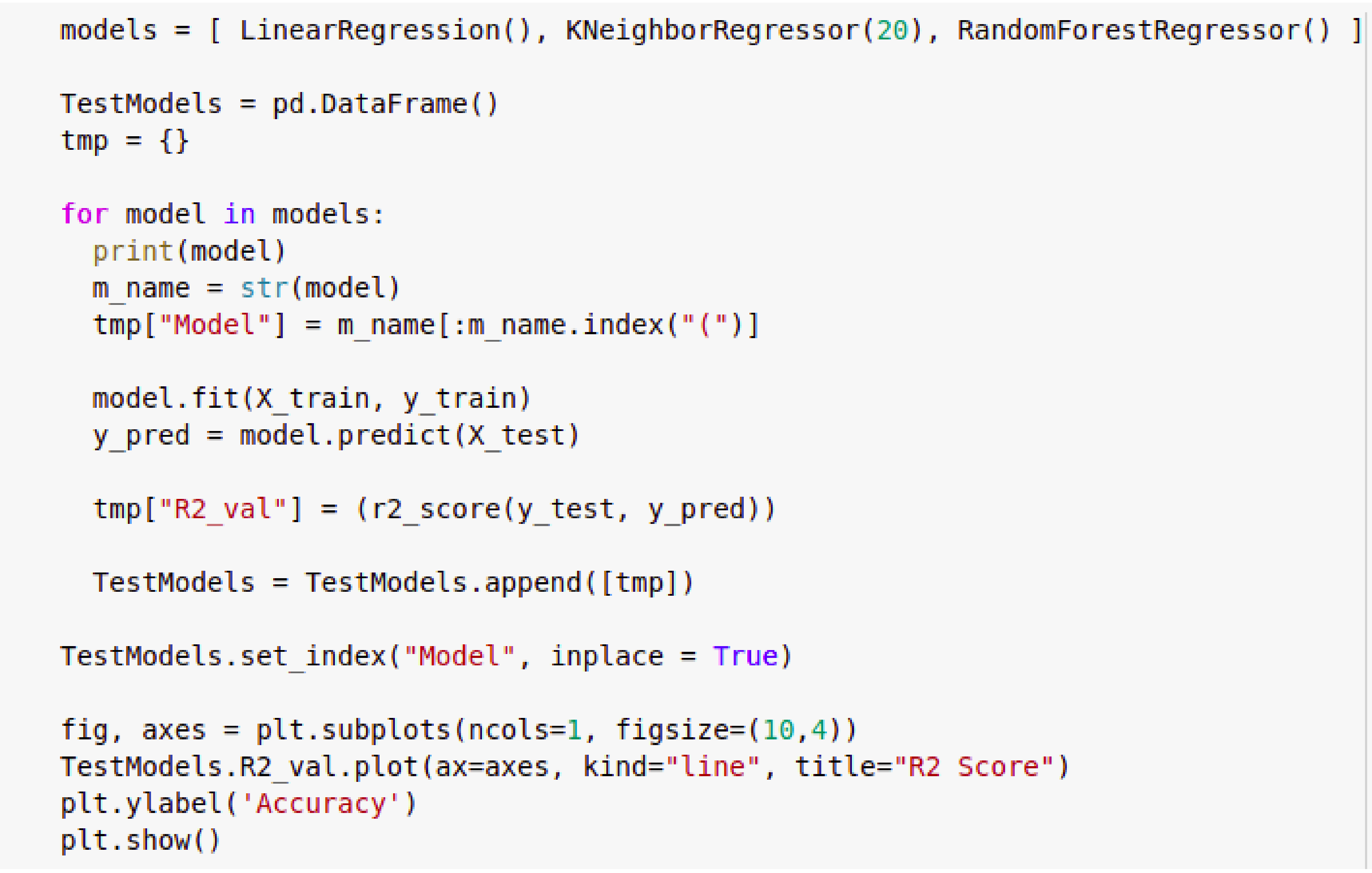
K from 10 to 30



Hence the required K value is between 17.5 and 20 and can be approximated to 19 or 20

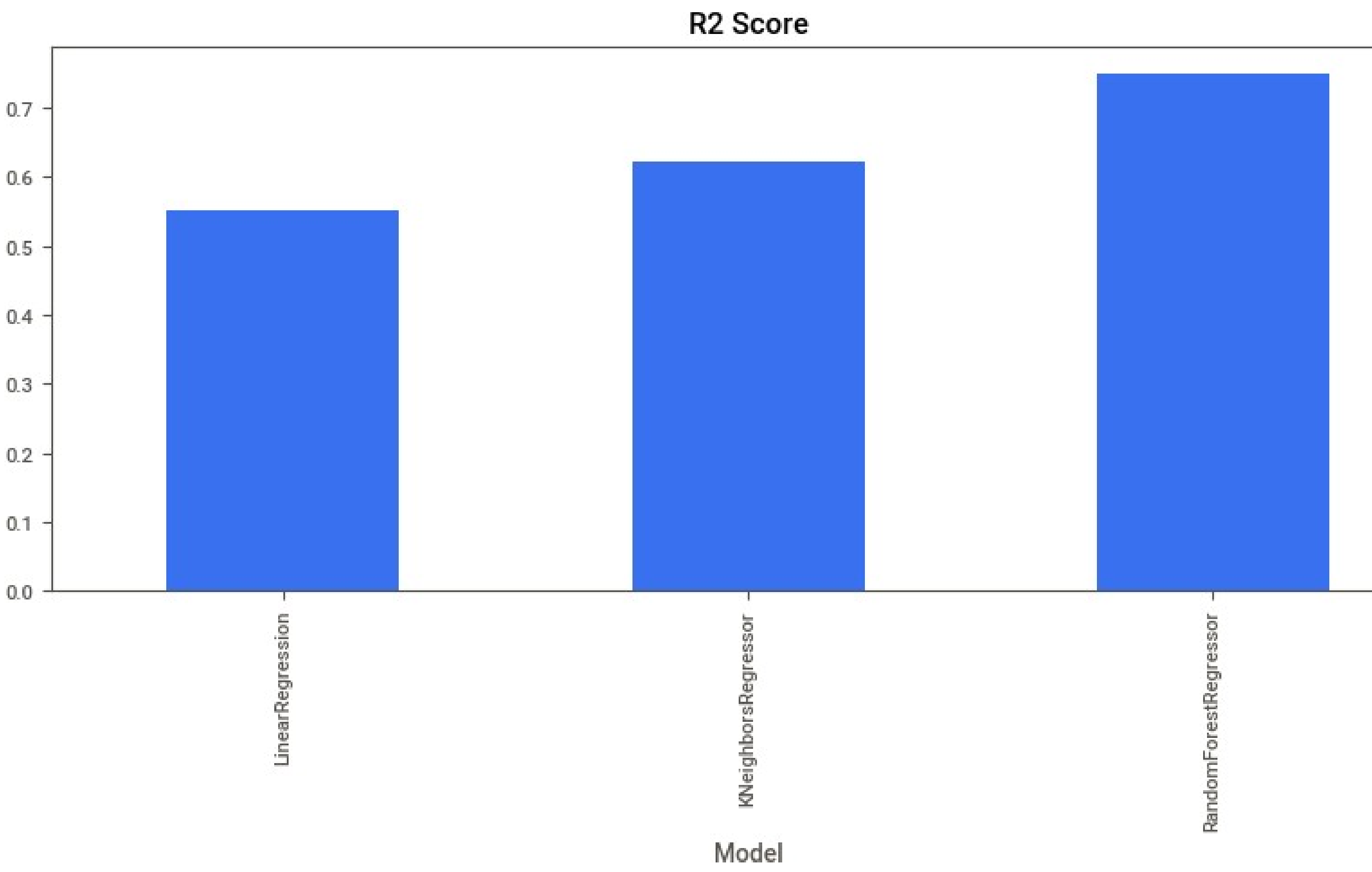
PLOTTING ACCURACIES OF ALL THREE ALGORITHMS

Here all the the three algorithms are evaluated and their repective accuracies are observed and plotted to determine the best algorithm out of the three

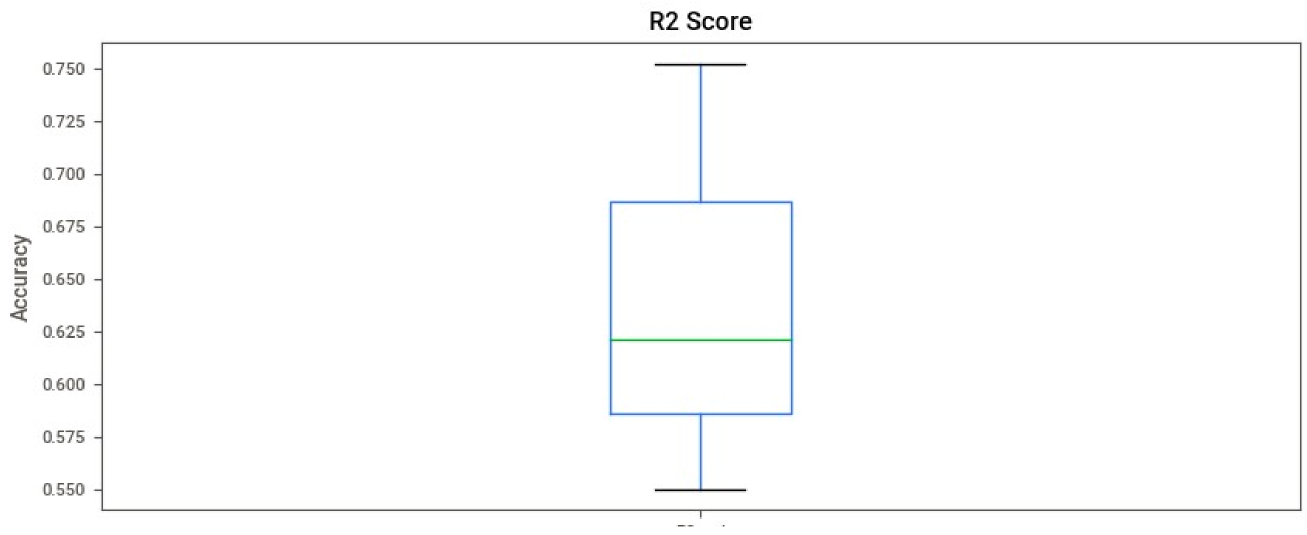


```
models = [ LinearRegression(), KNeighborRegressor(20), RandomForestRegressor() ]

TestModels = pd.DataFrame()
tmp = {}

for model in models:
  print(model)
  m_name = str(model)
  tmp["Model"] = m_name[:m_name.index("(")]

  model.fit(X_train, y_train)
  y_pred = model.predict(X_test)

  tmp["R2_val"] = (r2_score(y_test, y_pred))

  TestModels = TestModels.append([tmp])

TestModels.set_index("Model", inplace = True)

fig, axes = plt.subplots(ncols=1, figsize=(10,4))
TestModels.R2_val.plot(ax=axes, kind="line", title="R2 Score")
plt.ylabel('Accuracy')
plt.show()
```

Bar Graph depicting the most accurate algorithm for the model

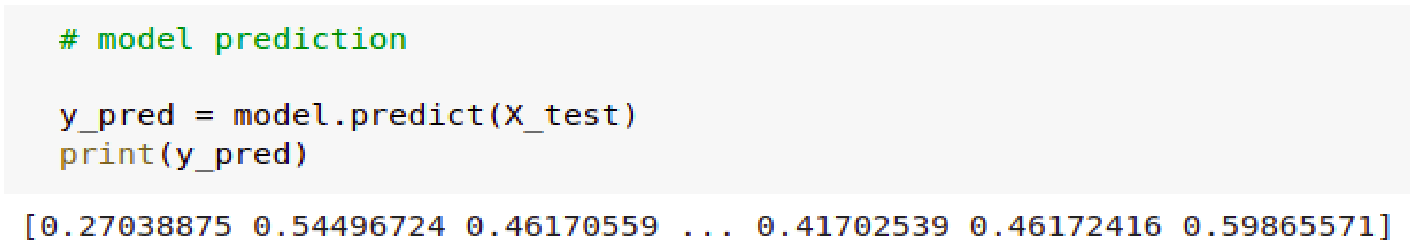


Box plot of accuracies and algorithms



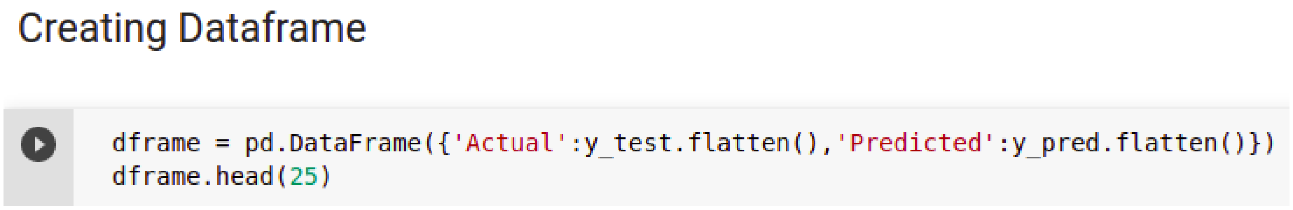
Since highest accuracy is of Random Forest we predict the values using Random forest itself

PREDICTING THE VALUES



```
# model prediction

y_pred = model.predict(X_test)
print(y_pred)
```

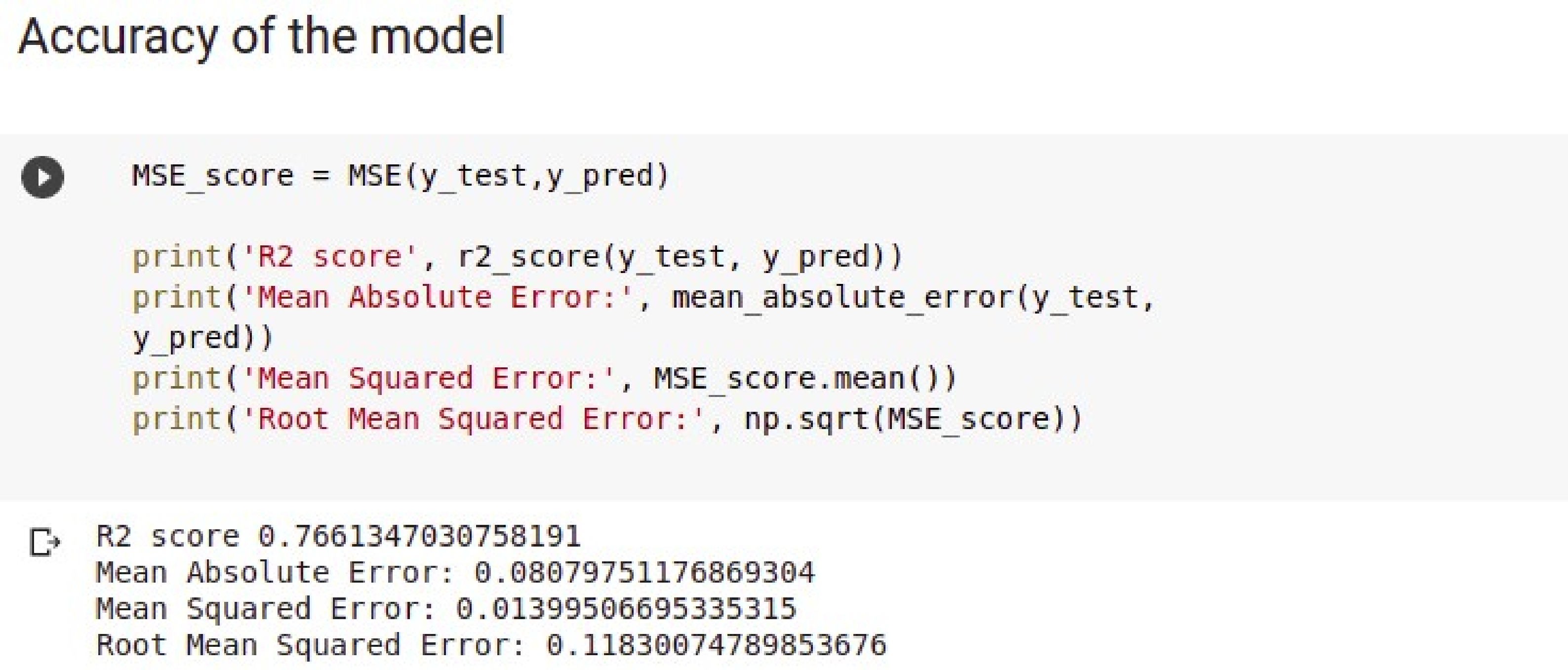
```
[0.27038875 0.54496724 0.46170559 ... 0.41702539 0.46172416 0.59865571]
```



Creating Dataframe

```
dframe = pd.DataFrame({'Actual':y_test.flatten(),'Predicted':y_pred.flatten()})
dframe.head(25)
```

MODEL ACCURACY

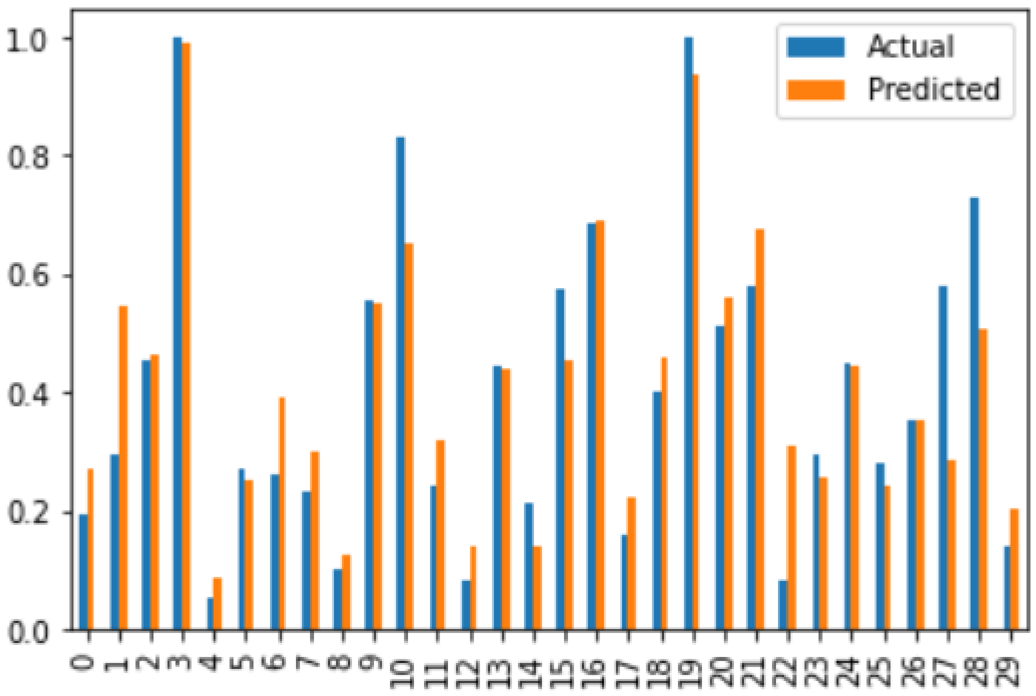


Accuracy of the model

```
MSE_score = MSE(y_test,y_pred)

print('R2 score', r2_score(y_test, y_pred))
print('Mean Absolute Error:', mean_absolute_error(y_test,
y_pred))
print('Mean Squared Error:', MSE_score.mean())
print('Root Mean Squared Error:', np.sqrt(MSE_score))
```

```
R2 score 0.7661347030758191
Mean Absolute Error: 0.08079751176869304
Mean Squared Error: 0.013995066953353l5
Root Mean Squared Error: 0.11830074789853676
```

Visualizing our model's predictions


Actual
Predicted