EARTHQUAKE PREDICTION MODEL USING PYTHON

Phase 3 Submission Document:

Project: Earthquake_Prediction

Phase 3: Development Part 1

INTRODUCTION:

✓ Earthquake Prediction is a way of predicting the magnitude of an earthquake based on parameters such as longitude, latitude, depth, and duration magnitude, country, and depth using machine learning to give warnings of potentially damaging earth quakes early enough to allow appropriate response to the disaster, Enabling people to minimize loss of life and property**.**

1.Feature Engineering:

1. Extracting meaningful features from the data, such as temporal trends, spatial relationships, and interactions between different parameters.

2. Incorporating domain-specific knowledge and external factors that may influence seismic activities.

2.Machine Learning Models:

1. Developing predictive models, often based on machine learning algorithms, to analyze and learn patterns from historical earthquake data.

2. Common models include neural networks, support vector machines, and decision trees.

3. Hyperparameter Tuning:

1. Fine-tuning model parameters to optimize predictive performance through techniques like grid search or Bayesian optimization.

4. Validation and Evaluation:

1. Splitting the dataset into training and testing sets to validate the model's performance.

2. Evaluating the model's accuracy, precision, recall, and other relevant metrics.

5. Challenges in Earthquake Prediction:

1. Earthquakes are inherently unpredictable due to the dynamic and complex nature of tectonic processes.

2. Limited historical data for rare, large-magnitude earthquakes makes it challenging to train accurate models.

Developmet Part 1:

- **Loading**
- **Preprocessing**

Loading DataSet:

Data loading in earthquake prediction is a crucial step in the process. It involves gathering and organizing relevant information to train and test predictive models

Downloading DataSet:

https://www.kaggle.com/datasets/usgs/earthquake-database

Preprocessing Data:

Preprocessing is a crucial step in earthquake prediction asit helps clean and transform raw data into a format suitable for training machine learning models.

Handling Missing Data:

Check for missing values in your dataset decideon a strategy to handle them. You can either remove rows with missing values, fill themusing imputation techniques, or use more advanced methodsdepending on the nature of the missing data.

Program:

[1]:

```
import numpy as np import
pandas as pd import requests
from sklearn import preprocessing import
matplotlib.pyplot as plt import seaborn as
sns
from pandas.plotting import scatter_matrix
from sklearn.impute import SimpleImputer
from sklearn.preprocessing importStandardScaler from
sklearn.preprocessing import MinMaxScaler import time
```

[2]:

```
df =pd.read_csv("database.csv")
```

[3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23412 entries, 0 to 23411
Data   columns (total 21 columns):
 #     Column                        Non-Null Count    Dtype
--------------------------------------------------------    ----------
 0     Date                          23412 non-null    object
 1     Time                          23412 non-null    object
 2     Latitude                      23412 non-null    float64
 3     Longitude                     23412 non-null    float64
 4     Type                          23412 non-null    object
 5     Depth                         23412 non-null    float64
 6     Depth Error                   4461 non-null     float64
 7     Depth Seismic Stations        7097 non-null     float64
 8     Magnitude                     23412 non-null    float64
 9     Magnitude Type                23409 non-null    object
 10    Magnitude Error               327 non-null      float64
 11    Magnitude Seismic Stations    2564 non-null     float64
 12    Azimuthal Gap                 7299 non-null     float64
 13    Horizontal Distance           1604 non-null     float64
 14    Horizontal Error              1156 non-null     float64
 15    RootMeanSquare                17352 non-null    float64
 16    ID                            23412 non-null    object
```

```
 17   Source                23412 non-null   object
 18   Location Source       23412 non-null   object
 19   Magnitude Source      23412 non-null   object
 20   Status                23412 non-null   object
dtypes: float64(12), object(9)
memory usage: 3.8+ MB
```

[4]:

```
df.describe()
```

[4]:

	Latitude	Longitude	Depth	Depth Error
count	23412.000000	23412.000000	23412.000000	4461.000000
mean	1.679033	39.639961	70.767911	4.993115
std	30.113183	125.511959	122.651898	4.875184
min	-77.080000	-179.997000	-1.100000	0.000000
25%	-18.653000	-76.349750	14.522500	1.800000
50%	-3.568500	103.982000	33.000000	3.500000
75%	26.190750	145.026250	54.000000	6.300000
max	86.005000	179.998000	700.000000	91.295000

	Depth Seismic Stations	Magnitude	Magnitude Error
count	7097.000000	23412.000000	327.000000
mean	275.364098	5.882531	0.071820
std	162.141631	0.423066	0.051466
min	0.000000	5.500000	0.000000
25%	146.000000	5.600000	0.046000
50%	255.000000	5.700000	0.059000
75%	384.000000	6.000000	0.075500
max	934.000000	9.100000	0.410000

	Magnitude Seismic Stations	Azimuthal Gap	Horizontal Distance
count	2564.000000	7299.000000	1604.000000
mean	48.944618	44.163532	3.992660
std	62.943106	32.141486	5.377262
min	0.000000	0.000000	0.004505
25%	10.000000	24.100000	0.968750
50%	28.000000	36.000000	2.319500
75%	66.000000	54.000000	4.724500
max	821.000000	360.000000	37.874000

	Horizontal Error	RootMeanSquare
count	1156.000000	17352.000000
mean	7.662759	1.022784
std	10.430396	0.188545
min	0.085000	0.000000
25%	5.300000	0.900000
50%	6.700000	1.000000
75%	8.100000	1.130000

```
max              99.000000            3.440000
```

[5]:
```
df.shape
```

[5]: (23412, 21)

[6]:
```
df.head()
```

[6]:
```
          Date      Time  Latitude  Longitude        Type  Depth  Depth  Error  \
0   01/02/1965  13:44:18    19.246    145.616  Earthquake  131.6                NaN
1   01/04/1965  11:29:49     1.863    127.352  Earthquake   80.0                NaN
2   01/05/1965  18:05:58   -20.579   -173.972  Earthquake   20.0                NaN
3   01/08/1965  18:49:43   -59.076    -23.557  Earthquake   15.0                NaN
4   01/09/1965  13:32:50    11.938    126.427  Earthquake   15.0                NaN

    Depth Seismic Stations   Magnitude Magnitude Type  ...  \
0                      NaN         6.0             MW  ...
1                      NaN         5.8             MW  ...
2                      NaN         6.2             MW  ...
3                      NaN         5.8             MW  ...
4                      NaN         5.8             MW  ...
    Magnitude Seismic Stations   Azimuthal Gap   Horizontal Distance   \
0                          NaN             NaN                   NaN
1                          NaN             NaN                   NaN
2                          NaN             NaN                   NaN
3                          NaN             NaN                   NaN
4                          NaN             NaN                   NaN

    Horizontal Error   RootMeanSquare            ID  Source Location Source   \
0                NaN              NaN  ISCGEM860706  ISCGEM          ISCGEM
1                NaN              NaN  ISCGEM860737  ISCGEM          ISCGEM
2                NaN              NaN  ISCGEM860762  ISCGEM          ISCGEM
3                NaN              NaN  ISCGEM860856  ISCGEM          ISCGEM
4                NaN              NaN  ISCGEM860890  ISCGEM          ISCGEM

   Magnitude Source      Status
0            ISCGEM   Automatic
1            ISCGEM   Automatic
2            ISCGEM   Automatic
3            ISCGEM   Automatic
4            ISCGEM   Automatic

[5 rows x 21 columns]
```

[7]:
```
df.columns
```

[7] : Index(['Date', 'Time', 'Latitude', 'Longitude', 'Type', 'Depth', 'DepthError', 'Depth Seismic Stations',
'Magnitude', 'Magnitude Type',
'Magnitude Error', 'Magnitude Seismic Stations', 'Azimuthal Gap', 'Horizontal Distance',
'Horizontal Error', 'RootMeanSquare', 'ID', 'Source', 'Location Source', 'Magnitude Source',
'Status'], dtype='object')

[8]:
```
df.dtypes
```

[8] :
```
Date                          object
Time                          object
Latitude                      float64
Longitude                     float64
Type                          object
Depth                         float64
Depth Error                   float64
Depth Seismic Stations        float64
Magnitude                     float64
Magnitude Type                object
Magnitude Error               float64
Magnitude Seismic Stations    float64
Azimuthal Gap                 float64
Horizontal Distance           float64
Horizontal Error              float64
RootMeanSquare                float64
ID                            object
Source                        object
Location Source               object
Magnitude Source              object
Status                        object
dtype: object
```

[9]:
```
label_encoder = preprocessing.LabelEncoder()
for col in df.columns:
    if df[col].dtype == 'object':
        label_encoder.fit(df[col])
        df[col] =label_encoder.transform(df[col]) df.dtypes
```

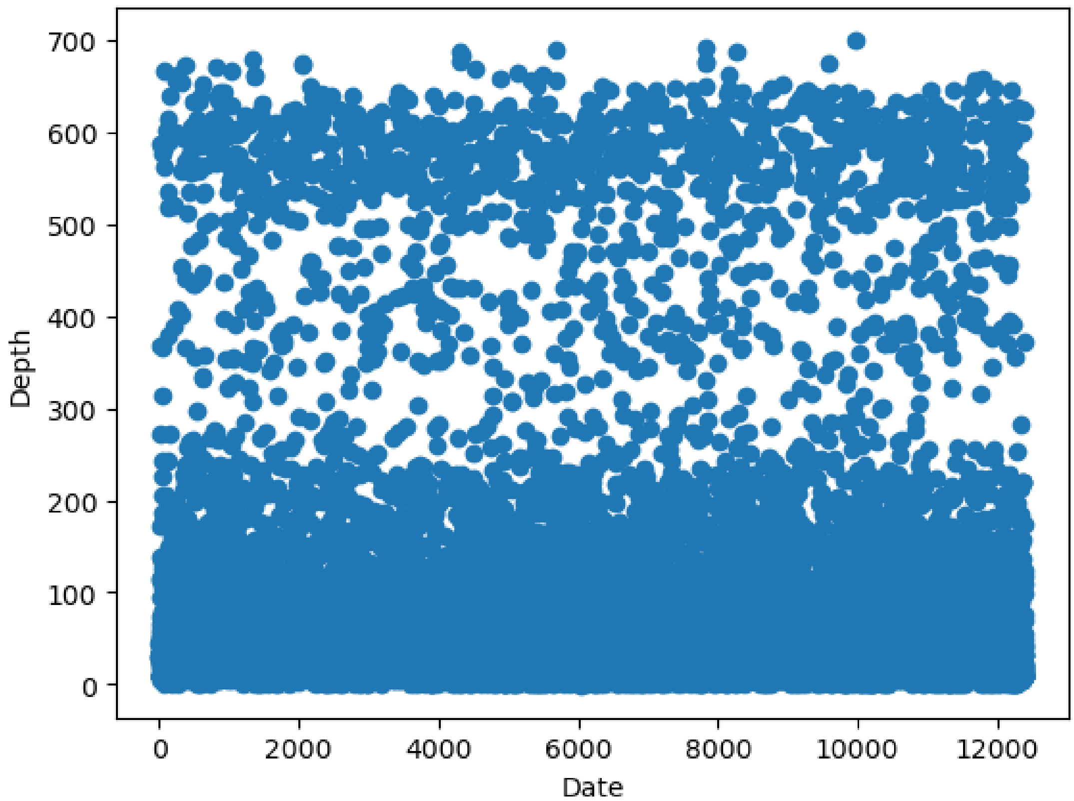
[9] :
```
Date                          int32
Time                          int32
Latitude                      float64
Longitude                     float64
Type                          int32
Depth                         float64
Depth Error                   float64
Depth Seismic Stations        float64
```

```
Magnitude                    float64
Magnitude Type                 int32
Magnitude Error              float64
Magnitude  Seismic  Stations  float64  Azimuthal
Gap                          float64
Horizontal Distance          float64
Horizontal Error             float64
RootMeanSquare               float64
ID                             int32
Source                         int32
Location Source                int32
Magnitude Source               int32
Status                         int32
dtype: object
```

[10]:
```
df.isnull().sum()
```

[10] :
```
Date                              0
Time                              0
Latitude                          0
Longitude                         0
Type                              0
Depth                             0
Depth Error                   18951
Depth Seismic Stations        16315
Magnitude                         0
Magnitude Type                    0
Magnitude Error               23085
Magnitude Seismic Stations    20848
Azimuthal Gap                 16113
Horizontal Distance           21808
Horizontal Error              22256
RootMeanSquare                 6060
ID                                0
Source                            0
Location Source                   0
Magnitude Source                  0
Status                            0
dtype: int64
```

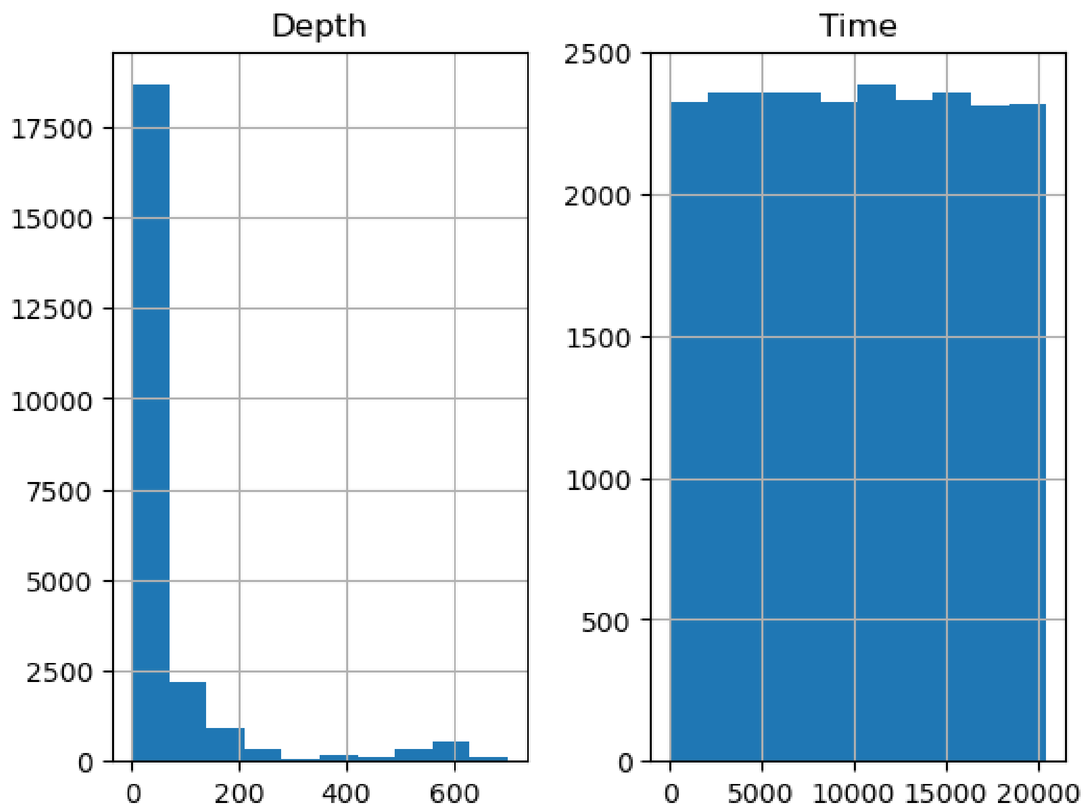
[11]:
```
plt.scatter(df.Date, df.Depth)
plt.xlabel('Date') plt.ylabel("Depth")
plt.show()
```



[12]:

```
plt.figure(figsize=(7,7))
df[['Depth','Time']].hist() plt.show()
```

<Figure size 700x700 with 0 Axes>



[13]:
```
import plotly.express as px
px.scatter(df, x='Date',y='Depth', color="Depth Error")
```

[14]:
```
most_correlated =df.corr()['Depth'].sort_values(ascending=False) most_correlated
```

[14]:
```
Depth                         1.000000
Depth Seismic Stations        0.174663
Magnitude                     0.023457
Location Source               0.022434
Source                        0.012838
Time                          0.010643
Status                        0.003848
Date                          0.002878
ID                           -0.001201
Magnitude Source             -0.013492
Magnitude   Seismic   Stations   -0.015254
Horizontal Error             -0.016467
Magnitude Type               -0.024904
Type                         -0.050394
Horizontal Distance          -0.073832
```

```
Depth Error                  -0.074609
Magnitude Error              -0.076918
Latitude                     -0.081020
Longitude                    -0.085861
RootMeanSquare               -0.134002
Azimuthal Gap                -0.171162
Name: Depth, dtype: float64
```

[15]:

```
scaler = preprocessing.MinMaxScaler() d =
scaler.fit_transform(df)
df = pd.DataFrame(d,columns=df.columns)
df.head()
```

[15]:

```
       Date      Time  Latitude  Longitude  Type     Depth  Depth Error  \
0  0.002742  0.575644  0.590649   0.904493   0.0  0.189274          NaN
1  0.008145  0.481901  0.484060   0.853759   0.0  0.115675          NaN
2  0.010726  0.754580  0.346451   0.016736   0.0  0.030096          NaN
3  0.018952  0.785843  0.110396   0.434562   0.0  0.022964          NaN
4  0.021532  0.567193  0.545838   0.851190   0.0  0.022964          NaN

   Depth Seismic Stations  Magnitude      Magnitude Type  ...  \
0                     NaN   0.138889                 0.5  ...
1                     NaN   0.083333                 0.5  ...
2                     NaN   0.194444                 0.5  ...
3                     NaN   0.083333                 0.5  ...
4                     NaN   0.083333                 0.5  ...

   Magnitude Seismic Stations     Azimuthal Gap   Horizontal Distance     \
0                          NaN              NaN                    NaN
1                          NaN              NaN                    NaN
2                          NaN              NaN                    NaN
3                          NaN              NaN                    NaN
4                    NaN   NaN                     NaN Horizontal Error

                     RootMeanSquare        ID    Source   Location Source   \
0                NaN            NaN  0.110162  0.333333          0.425532
1                NaN            NaN  0.110205  0.333333          0.425532
2                NaN            NaN  0.110247  0.333333          0.425532
3                NaN            NaN  0.110290  0.333333          0.425532
4                NaN            NaN  0.110333  0.333333          0.425532
   Magnitude Source    Status
 0         0.478261       0.0
 1         0.478261       0.0
 2         0.478261       0.0
 3         0.478261       0.0
 4         0.478261       0.0
```

[5 rows x 21 columns]

[16]:
```
y=np.array(df['Magnitude'])
X=np.array(df.drop('Magnitude',axis=1))
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=2)
```

[18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
import time
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ␣

   ↪random_state=42)
imputer = SimpleImputer(strategy='mean')
X_train =imputer.fit_transform(X_train) X_test =
imputer.transform(X_test) linear =
LinearRegression()
start1 = time.time()
linear.fit(X_train, y_train) end1 =
time.time()
ans1 =linear.predict(X_test) t1 =
end1 - start1
```

[19]:
```
accuracy1=linear.score(X_test,y_test)
print("Accuracy of Linear Regression model is:",accuracy1)
```

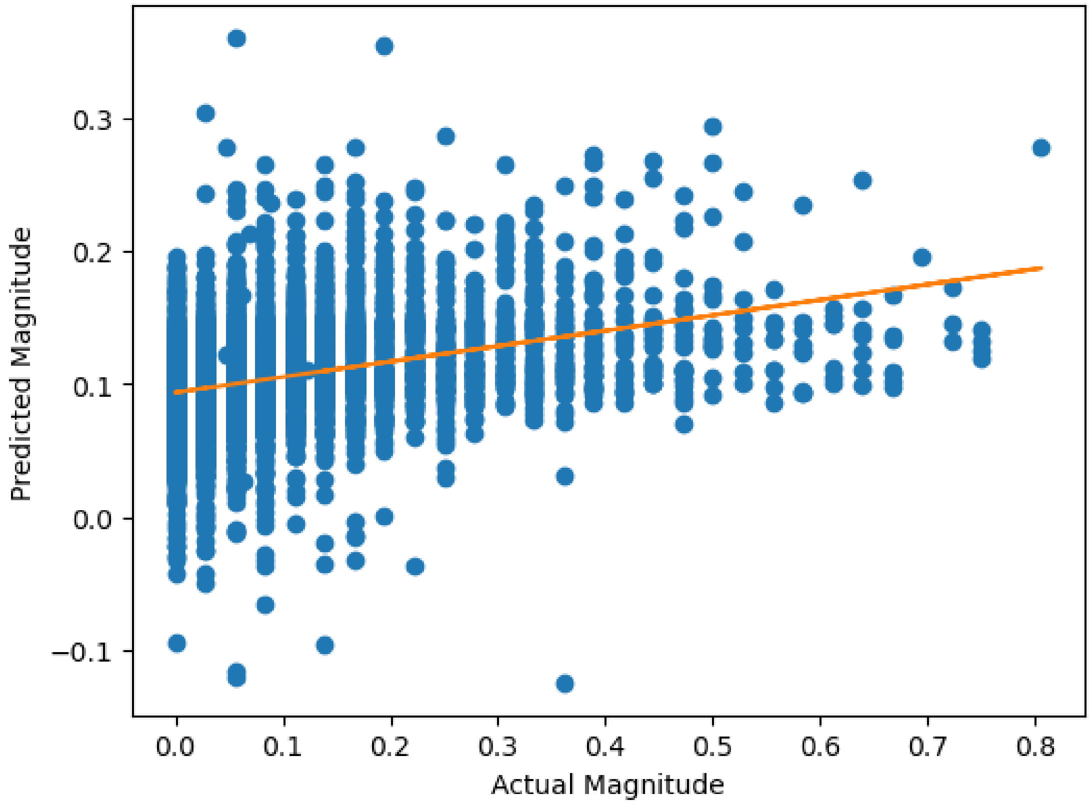
Accuracy of Linear Regression model is: 0.11252233011289892

[20]:
```
from sklearn import metrics print("Linear
Regression")
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, ans1)) print('Mean Squared
Error:', metrics.mean_squared_error(y_test, ans1)) print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, ␣

   ↪ans1)))
```

Linear Regression
Mean Absolute Error:0.08146087772759264 Mean
Squared Error: 0.0126305846857069
Root Mean Squared Error: 0.11238587404877402

[21]:
```
plt.plot(y_test, ans1, 'o')
m, b = np.polyfit(y_test,ans1, 1)
plt.plot(y_test, m*y_test + b)
plt.xlabel("Actual Magnitude")
plt.ylabel("Predicted Magnitude")
```

[21]: Text(0, 0.5, 'Predicted Magnitude')



[22]:
```
from sklearn.tree importDecisionTreeRegressor start2 =
time.time()
regressor = DecisionTreeRegressor(random_state = 40)
regressor.fit(X_train,y_train)
ans2 = regressor.predict(X_test) end2 =
time.time()
t2 = end2-start2
```

[23]:
```
accuracy2=regressor.score(X_test,y_test) print("Accuracy of Decision
Tree model is:",accuracy2)
```

Accuracy of Decision Tree model is: -0.47913164184956325

[24]:
```
print("Decision Tree")
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, ans2)) print('Mean Squared
Error:', metrics.mean_squared_error(y_test, ans2)) print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, ␣
  ↪ans2)))
```

Decision Tree

Mean Absolute Error: 0.09996381711628347 Mean
Squared Error: 0.021051005673265257 Root Mean
Squared Error:0.1450896470230225

[25]:
```
from sklearn.neighbors importKNeighborsRegressor start3 =
time.time()
knn =KNeighborsRegressor(n_neighbors=6)
knn.fit(X_train, y_train)
ans3 = knn.predict(X_test) end3 =
time.time()
t3 = end3-start3
```

[26]:
```
accuracy3=knn.score(X_test,y_test) print("Accuracy of
KNN modelis:",accuracy3)
```

Accuracy of KNN model is: 0.04140772305714546

[27]:
```
print("KNN Model")
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, ans3)) print('Mean Squared
Error:', metrics.mean_squared_error(y_test, ans3)) print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, ␣
  ↪ans3)))
```

KNN Model
Mean Absolute Error: 0.08315370409914505 Mean
Squared Error: 0.013642687972680563 Root Mean
Squared Error:0.1168019176755269

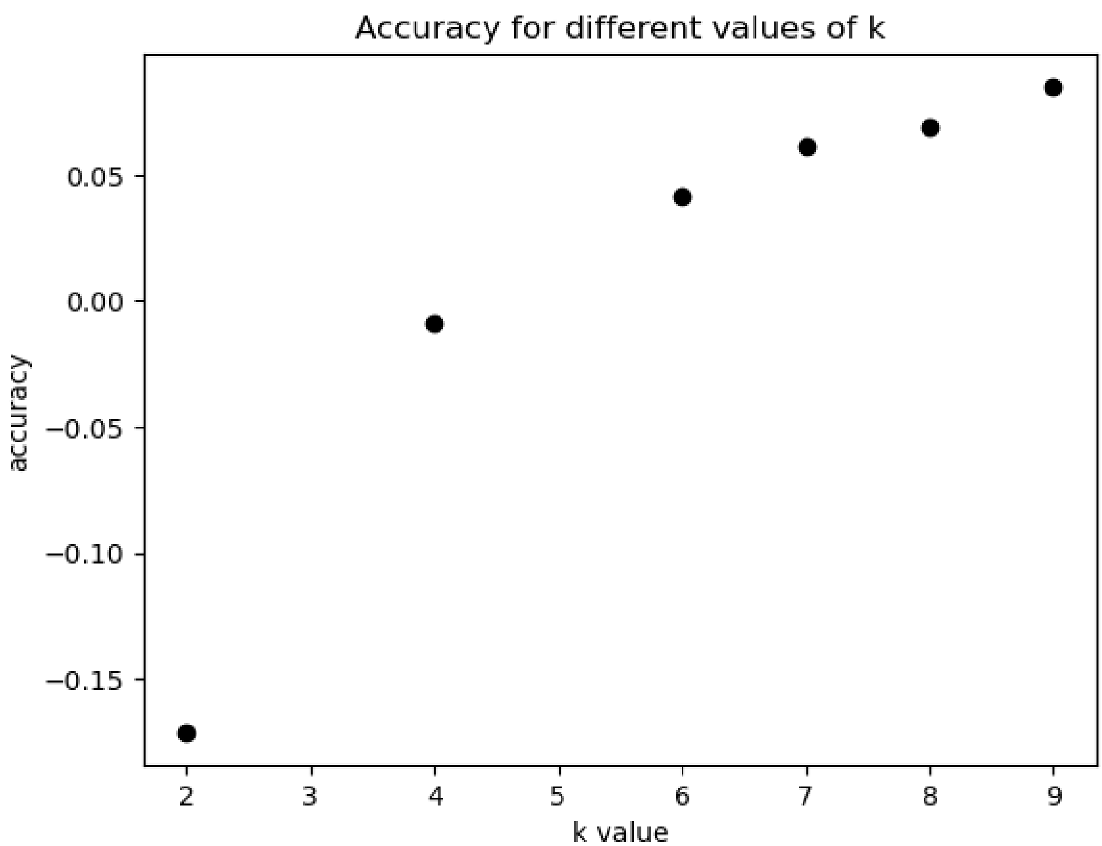
[28]:
```
import random
info = {}
for i in range(10):
    k = random.randint(2,10) startk =
    time.time()
    knn =KNeighborsRegressor(n_neighbors=k)
    knn.fit(X_train, y_train)
    ans3 = knn.predict(X_test) endk =
    time.time()
    tk = endk-startk
    acc3=knn.score(X_test,y_test) info[k] =
    [acc3,tk]

for i in info:
    print("for k =",i,": accuracy =",info[i][0])
```

accuracy =-0.009107726353721368 for k = 9 : accuracy
= 0.08499315529333218 for k = 2 : accuracy = -
0.17114696576440624

for k = 6 : accuracy =0.04140772305714546 for k = 7 : accuracy =0.06141803646878197

[29]:
```
x = list(info.keys()) yacc = []
for i in info: yacc.append(info[i][0])
plt.plot(x, yacc, 'o', color='black'); plt.xlabel("k
value") plt.ylabel("accuracy");
plt.title("Accuracy for different values of k")
```

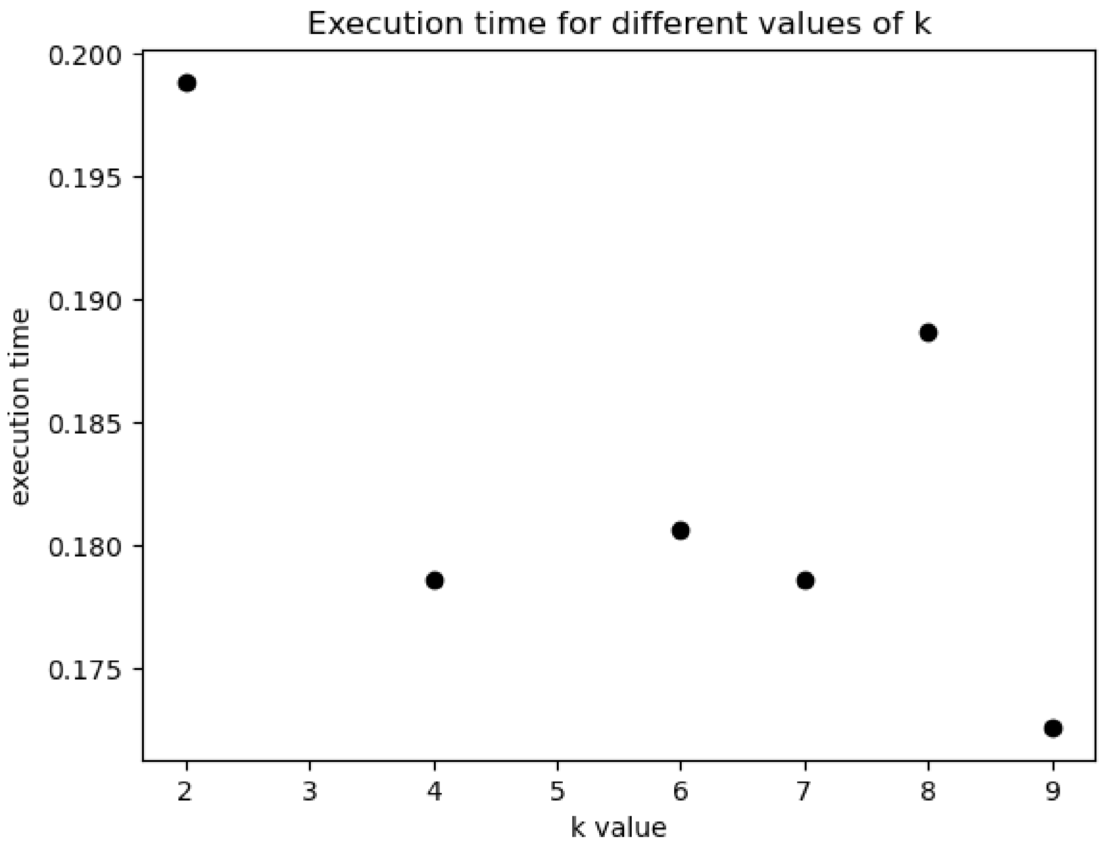
[29] : Text(0.5, 1.0, 'Accuracy for different values of k')



[30]:
```
yt = []
for i in info:
    yt.append(info[i][1])
plt.plot(x, yt, 'o', color='black'); plt.xlabel("k
value") plt.ylabel("execution time");
```

```
plt.title("Execution time for different values ofk")
```

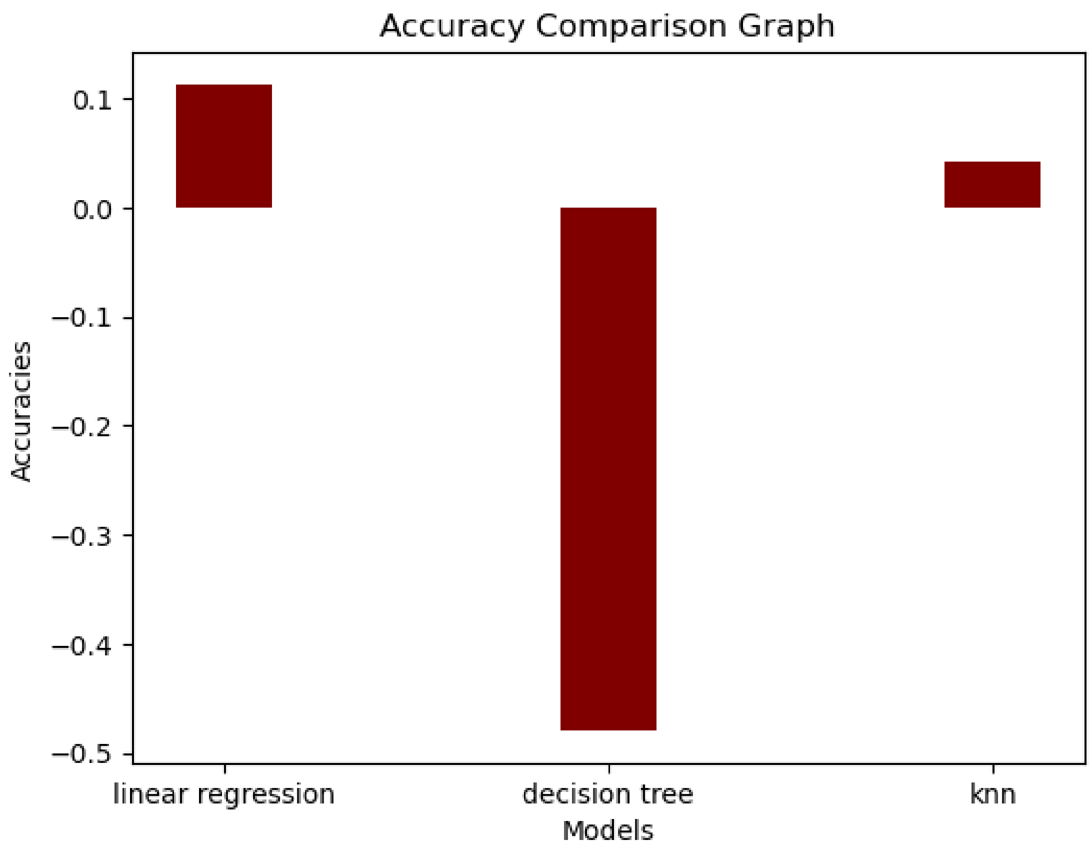
[30] : Text(0.5, 1.0, 'Execution time for different values of k')



[31]:
```
models = ["linear regression","decision tree","knn"] accuracies =
[accuracy1,accuracy2,accuracy3]
```

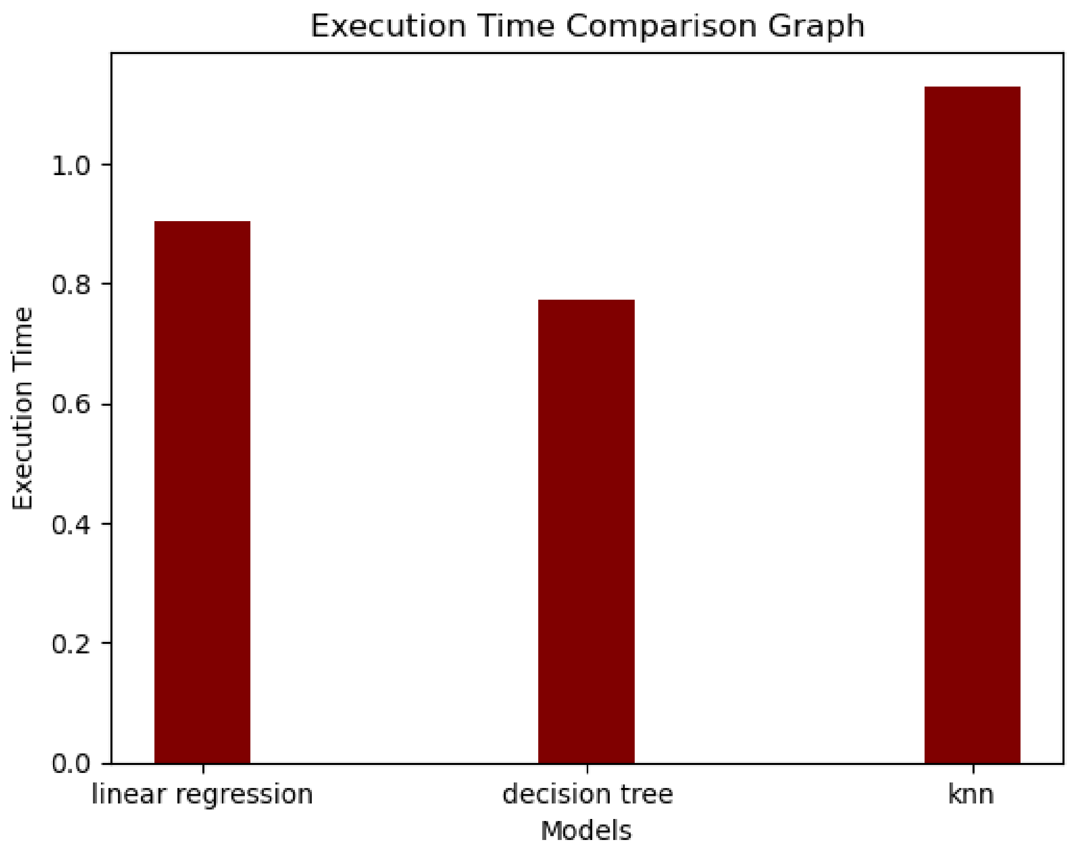
[32]:
```
plt.bar(models, accuracies, color ='maroon',width = 0.25)
plt.xlabel("Models")
plt.ylabel("Accuracies") plt.title("Accuracy
Comparison Graph")
```

[32] : Text(0.5, 1.0, 'Accuracy Comparison Graph')



[33]:
```
times = [t1,t2,t3]
plt.bar(models, times, color='maroon', width =
            0.25)
plt.xlabel("Models") plt.ylabel("Execution
Time")
plt.title("Execution Time Comparison Graph")
```

[33] : Text(0.5, 1.0, 'Execution Time Comparison Graph')



[ ]:

CONCLUSION:

conclusion, the development of an earthquake prediction machine learning model using Python involves a systematic and multidimensional approach. Here's an elaborative summary of key aspects: In summary, the development of an earthquake prediction model using Python is a multifaceted process that requires a combination of domain knowledge, data science expertise, and continuous improvement. Through careful preprocessing, model development, and collaboration, Python serves as a versatile tool for addressing the complexities of earthquake prediction and contributing to advancements in the field