# Final Assessment 1

Kaviyadevi(20106064)

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
#importing dataset
data=pd.read_csv(r"C:\Users\user\Downloads\stations.csv")
data
```

Out[4]:

| | id | name | address | lon | lat | elevation |
|---|---|---|---|---|---|---|
| 0 | 28079004 | Pza. de España | Plaza de España | -3.712247 | 40.423853 | 635 |
| 1 | 28079008 | Escuelas Aguirre | Entre C/ Alcalá y C/ O' Donell | -3.682319 | 40.421564 | 670 |
| 2 | 28079011 | Avda. Ramón y Cajal | Avda. Ramón y Cajal esq. C/ Príncipe de Vergara | -3.677356 | 40.451475 | 708 |
| 3 | 28079016 | Arturo Soria | C/ Arturo Soria esq. C/ Vizconde de los Asilos | -3.639233 | 40.440047 | 693 |
| 4 | 28079017 | Villaverde | C/. Juan Peñalver | -3.713322 | 40.347139 | 604 |
| 5 | 28079018 | Farolillo | Calle Farolillo - C/Ervigio | -3.731853 | 40.394781 | 630 |
| 6 | 28079024 | Casa de Campo | Casa de Campo (Terminal del Teleférico) | -3.747347 | 40.419356 | 642 |
| 7 | 28079027 | Barajas Pueblo | C/. Júpiter, 21 (Barajas) | -3.580031 | 40.476928 | 621 |
| 8 | 28079035 | Pza. del Carmen | Plaza del Carmen esq. Tres Cruces. | -3.703172 | 40.419208 | 659 |
| 9 | 28079036 | Moratalaz | Avd. Moratalaz esq. Camino de los Vinateros | -3.645306 | 40.407947 | 685 |
| 10 | 28079038 | Cuatro Caminos | Avda. Pablo Iglesias esq. C/ Marqués de Lema | -3.707128 | 40.445544 | 698 |
| 11 | 28079039 | Barrio del Pilar | Avd. Betanzos esq. C/ Monforte de Lemos | -3.711542 | 40.478228 | 674 |
| 12 | 28079040 | Vallecas | C/ Arroyo del Olivar esq. C/ Río Grande. | -3.651522 | 40.388153 | 677 |
| 13 | 28079047 | Mendez Alvaro | C/ Juan de Mariana / Pza. Amanecer Mendez Alvaro | -3.686825 | 40.398114 | 599 |
| 14 | 28079048 | Castellana | C/ Jose Gutierrez Abascal | -3.690367 | 40.439897 | 676 |
| 15 | 28079049 | Parque del Retiro | Paseo Venezuela- Casa de Vacas | -3.682583 | 40.414444 | 662 |
| 16 | 28079050 | Plaza Castilla | Plaza Castilla (Canal) | -3.688769 | 40.465572 | 728 |
| 17 | 28079054 | Ensanche de Vallecas | Avda La Gavia / Avda. Las Suertes | -3.612117 | 40.372933 | 627 |
| 18 | 28079055 | Urb. Embajada | C/ Riaño (Barajas) | -3.580747 | 40.462531 | 618 |
| 19 | 28079056 | Pza. Fernández Ladreda | Pza. Fernández Ladreda - Avda. Oporto | -3.718728 | 40.384964 | 604 |
| 20 | 28079057 | Sanchinarro | C/ Princesa de Eboli esq C/ Maria Tudor | -3.660503 | 40.494208 | 700 |
| 21 | 28079058 | El Pardo | Avda. La Guardia | -3.774611 | 40.518058 | 615 |
| 22 | 28079059 | Juan Carlos I | Parque Juan Carlos I (frente oficinas mantenim... | -3.609072 | 40.465250 | 660 |
| 23 | 28079060 | Tres Olivos | Plaza Tres Olivos | -3.689761 | 40.500589 | 715 |

In [6]:
```
data.info()
```

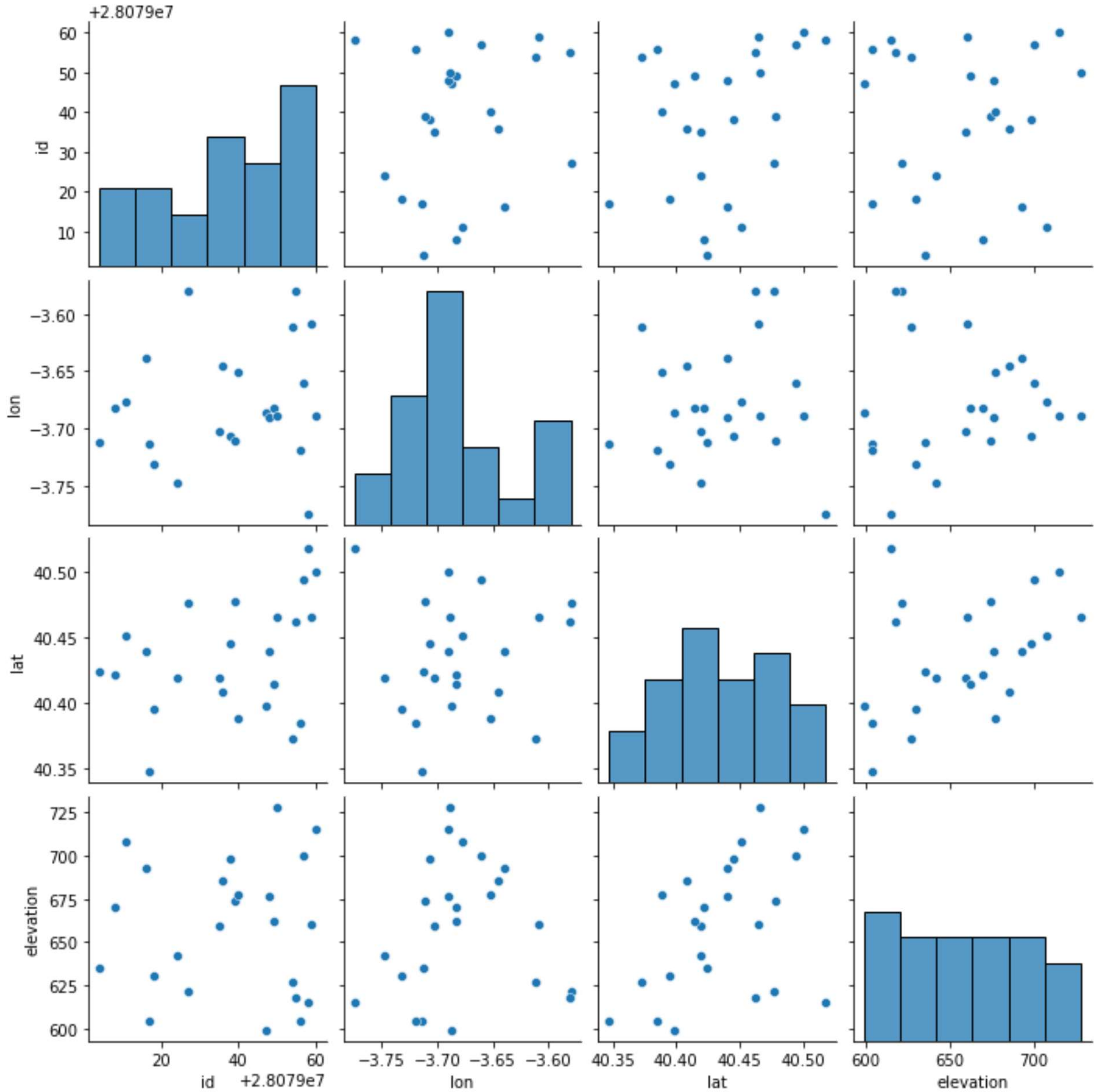
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   id         24 non-null     int64  
 1   name       24 non-null     object 
 2   address    24 non-null     object 
 3   lon        24 non-null     float64
 4   lat        24 non-null     float64
 5   elevation  24 non-null     int64  
dtypes: float64(2), int64(2), object(2)
memory usage: 1.2+ KB
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['id', 'name', 'address', 'lon', 'lat', 'elevation'], dtype='object')
```

In [9]:
```
sns.pairplot(df)
```

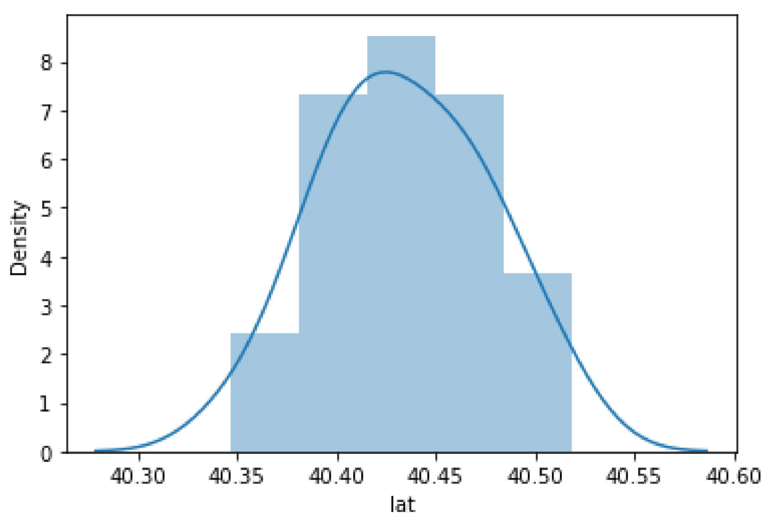
Out[9]: <seaborn.axisgrid.PairGrid at 0x1f561f42af0>



In [18]:
```
sns.distplot(data['lat'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[18]:
```
<AxesSubplot:xlabel='lat', ylabel='Density'>
```




# MODEL BUILDING

# 1.Linear Regression

In [11]:
```
df1=df[['id', 'lon', 'lat', 'elevation']]
```

In [26]:
```
x=df1[['id', 'lon', 'lat']]
y=df1[['elevation']]
```

In [27]:
```
#split the dataset into trainning and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [28]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

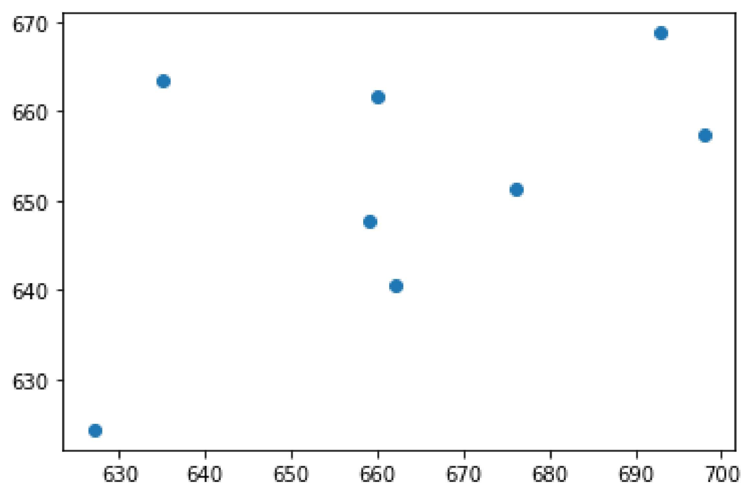
Out[28]:
```
LinearRegression()
```

In [29]:
```
print(lr.intercept_)
```

```
[12778338.31468202]
```

In [30]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x1f564ac38e0>
```




In [31]:
```
print(lr.score(x_test,y_test))
```

```
0.033234207960623596
```

# 2.Ridge Regression

In [32]:
```
from sklearn.linear_model import Ridge
```

In [33]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[33]:
```
Ridge(alpha=10)
```

In [34]:
```
rr.score(x_test,y_test)
```

Out[34]:
```
-0.1321146337718111
```

# 3.Lasso Regression

In [35]:
```
from sklearn.linear_model import Lasso
```

In [36]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[36]:
```
Lasso(alpha=10)
```

In [37]:
```
la.score(x_test,y_test)
```

Out[37]:
```
-0.1277802223297897
```

# 4.ElasticNet Regression

In [38]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[38]:
```
ElasticNet()
```

In [39]:
```
print(en.coef_)
```

```
[0.09020638 0.         0.55137415]
```

In [40]:
```
print(en.predict(x_test))
```

```
[653.66972253 657.56249315 655.37215378 656.63241645 657.06056021
 652.57831674 655.65729398 656.55624407]
```

In [41]:
```
print(en.score(x_test,y_test))
```

```
-0.13260909664647325
```

# 5.Logistic Regression

In [42]:
```
from sklearn.linear_model import LogisticRegression
```

In [49]:
```
feature_matrix = df1.iloc[:,0:6]
target_vector = df1.iloc[:,-1]
```

In [50]:
```
feature_matrix.shape
```

Out[50]:
```
(24, 4)
```

In [51]:
```
target_vector.shape
```

Out[51]:
```
(24,)
```

In [52]:
```
from sklearn.preprocessing import StandardScaler
```

In [53]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [54]:
```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

Out[54]:
```
LogisticRegression()
```

In [57]:
```
observation=[[1,2,3,4]]
```

In [58]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[715]
```

In [59]:
```
logr.classes_
```

Out[59]:
```
array([599, 604, 615, 618, 621, 627, 630, 635, 642, 659, 660, 662, 670,
       674, 676, 677, 685, 693, 698, 700, 708, 715, 728], dtype=int64)
```

In [60]:
```
logr.score(fs,target_vector)
```

Out[60]:
```
0.8333333333333334
```

# 6.Random Forest

In [63]:
```
df1=df[['id', 'lon', 'lat', 'elevation']]
x=df1[['id', 'lon', 'lat']]
y=df1[['elevation']]
```

In [64]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [65]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

```
<ipython-input-65-6b9282c2a062>:3: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  rfc.fit(x_train,y_train)
```

Out[65]:
```
RandomForestClassifier()
```

In [66]:
```
parameters = {'max_depth':[1,2,3,4,5],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]}
```

In [67]:
```
from sklearn.model_selection import GridSearchCV

grid_search =  GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring='acc
grid_search.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:
666: UserWarning: The least populated class in y has only 1 members, which is
less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:593: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:593: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:593: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)
```

In [68]:
```
grid_search.best_score_
```

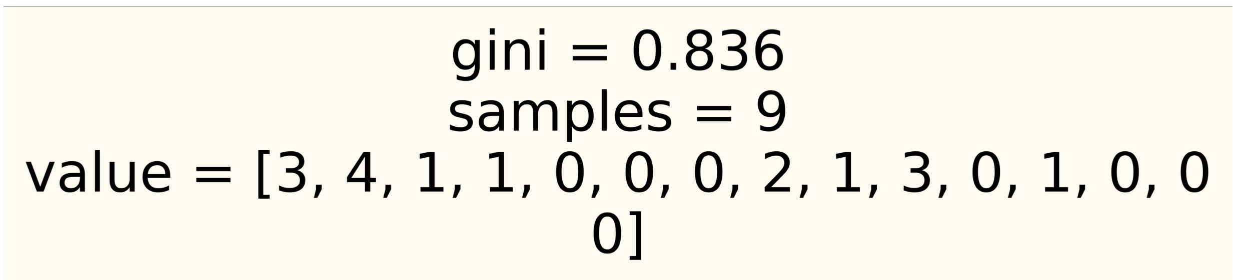
Out[68]:
```
0.125
```

In [69]:
```
rfc_best = grid_search.best_estimator_
```

In [70]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

Out[70]:
```
[Text(2232.0, 1087.2, 'gini = 0.836\nsamples = 9\nvalue = [3, 4, 1, 1, 0, 0, 0,
2, 1, 3, 0, 1, 0, 0\n0]')]
```




# Results

```
1.Linear regression : 0.033234207960623596

2.Ridge regression : -0.1321146337718111

3.Lasso regression : -0.1321146337718111

4.Elasticnet regression : -0.13260909664647325

5.Logistic regresssion : 0.8333333333333334

6.Random forest regression : 0.125

Hence Logistic regression gives high accuracy for this model.
```