In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as py
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
df=pd.read_csv(r"D:\New folder\stations.csv")
df
```

Out[2]:

	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

	id	name	address	lon	lat	elevation
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 [3]:
```
df.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 [4]:
```
df1=df.fillna(value=0)
df1
```

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

	id	name	address	lon	lat	elevation
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 [5]:
```
df1.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 [6]:
```
df1.columns
```

Out[6]:
```
Index(['id', 'name', 'address', 'lon', 'lat', 'elevation'], dtype='object')
```

In [9]:
```
df2=df1[['id','lon', 'lat', 'elevation']]
df2
```

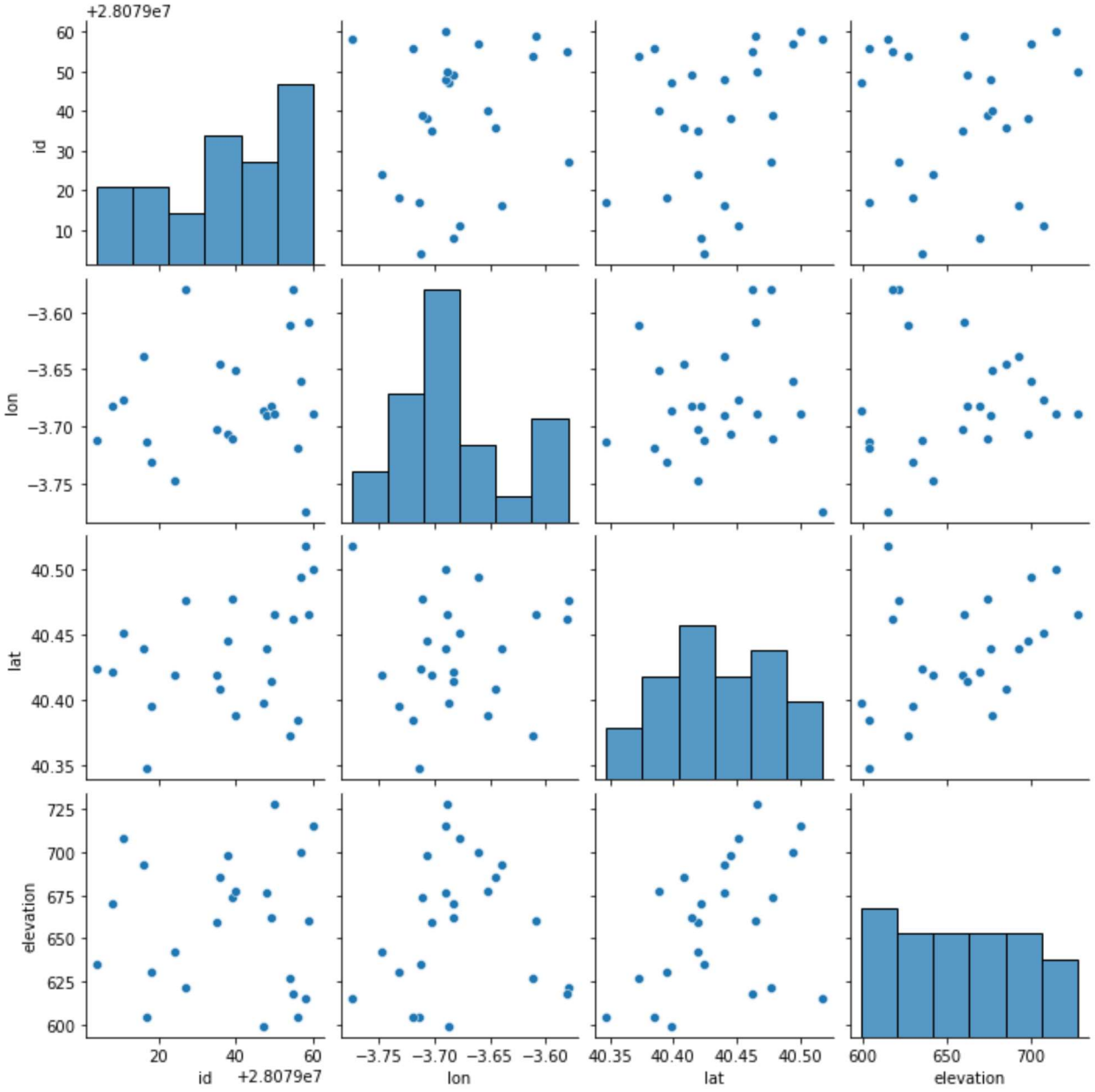
Out[9]:

	id	lon	lat	elevation
0	28079004	-3.712247	40.423853	635
1	28079008	-3.682319	40.421564	670

	id	lon	lat	elevation
2	28079011	-3.677356	40.451475	708
3	28079016	-3.639233	40.440047	693
4	28079017	-3.713322	40.347139	604
5	28079018	-3.731853	40.394781	630
6	28079024	-3.747347	40.419356	642
7	28079027	-3.580031	40.476928	621
8	28079035	-3.703172	40.419208	659
9	28079036	-3.645306	40.407947	685
10	28079038	-3.707128	40.445544	698
11	28079039	-3.711542	40.478228	674
12	28079040	-3.651522	40.388153	677
13	28079047	-3.686825	40.398114	599
14	28079048	-3.690367	40.439897	676
15	28079049	-3.682583	40.414444	662
16	28079050	-3.688769	40.465572	728
17	28079054	-3.612117	40.372933	627
18	28079055	-3.580747	40.462531	618
19	28079056	-3.718728	40.384964	604
20	28079057	-3.660503	40.494208	700
21	28079058	-3.774611	40.518058	615
22	28079059	-3.609072	40.465250	660
23	28079060	-3.689761	40.500589	715

In [10]:
```
sns.pairplot(df2)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1592dbfa550>
```

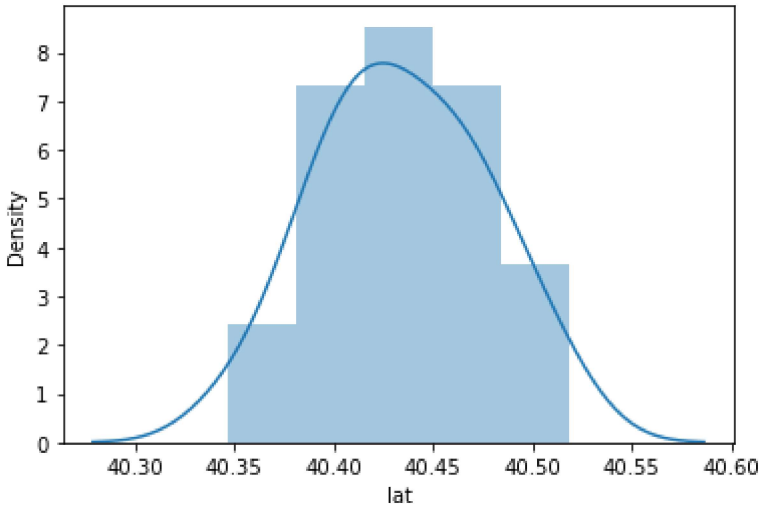


In [63]:

```
sns.distplot(df2['lat'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[63]: <AxesSubplot:xlabel='lat', ylabel='Density'>



In [65]:
```
x=df2[['id','lon', 'elevation']]
y=df2['lat']
```

In [66]:
```
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)
```

linear

In [67]:
```
from sklearn.linear_model import LinearRegression
```

In [68]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[68]:
```
LinearRegression()
```

In [69]:
```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[69]:

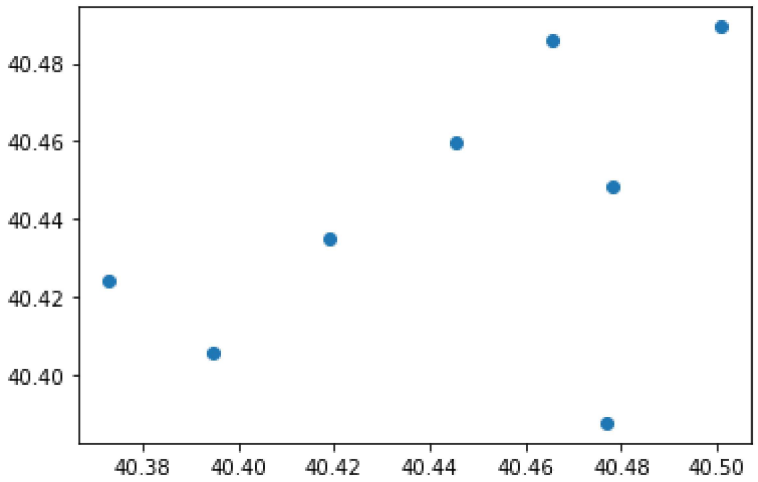
	Co-efficient
id	0.001053
lon	-0.148497
elevation	0.000538

In [70]:
```
print(lr.intercept_)
```

```
-29531.532685325794
```

In [71]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[71]: <matplotlib.collections.PathCollection at 0x1592f20f8e0>



In [72]:
```
print(lr.score(x_test,y_test))
```

0.10358598230866645

In [73]:
```
print(lr.score(x_train,y_train))
```

0.29562461158947695

Ridge

In [74]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [75]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[75]: Ridge(alpha=10)

In [76]:
```
rr.score(x_test,y_test)
```

Out[76]: 0.22146126524370047

Lasso

In [77]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[77]: Lasso(alpha=10)

In [78]:
```
la.score(x_test,y_test)
```

Out[78]: -0.11855214257829139

elasticnet

In [79]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[79]:
```
ElasticNet()
```

In [80]:
```
print(en.coef_)
```

```
[0. 0. 0.]
```

In [81]:
```
print(en.intercept_)
```

```
40.4298125
```

In [82]:
```
print(en.predict(x_test))
```

```
[40.4298125 40.4298125 40.4298125 40.4298125 40.4298125 40.4298125
 40.4298125 40.4298125]
```

In [83]:
```
print(en.score(x_test,y_test))
```

```
-0.11855214257829139
```

logistic

In [87]:
```
feature_matrix=df2.iloc[:,0:7]
target_vector=df2.iloc[:,-1]
```

In [113…
```
feature_matrix=df2[['id','lon', 'lat']]
y=df2['elevation']
```

In [114…
```
feature_matrix.shape
```

Out[114…
```
(24, 3)
```

In [115…
```
target_vector.shape
```

Out[115…
```
(24,)
```

In [116…
```
from sklearn.preprocessing import StandardScaler
```

In [117…
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [118…
```
logr =LogisticRegression()
logr.fit(fs,target_vector)
```

Out[118…
```
LogisticRegression()
```

In [119…
```
observation=[[1.4,2.3,5.0]]
```

In [120…
```
prediction=logr.predict(observation)
print(prediction)
```

```
[700]
```

In [121…
```
logr.classes_
```

Out[121…
```
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 [122…
```
logr.score(fs,target_vector)
```

Out[122…
```
0.5833333333333334
```

In [123…
```
logr.predict_proba(observation)[0][0]
```

Out[123…
```
0.00035567813727001925
```

In [124…
```
logr.predict_proba(observation)[0][1]
```

Out[124…
```
1.2261323316887416e-06
```

random forest

In [125…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
```

In [129…
```
x=df2.drop('elevation',axis=1)
y=df2['elevation']
```

In [130…
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.70)
```

In [131…
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[131…
```
RandomForestClassifier()
```

In [132…

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

In [133…

```
from sklearn.model_selection import GridSearchCV
```

In [134…

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:666: UserWa
rning: 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"
```

Out[134…

```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [135…

```
grid_search.best_score_
```

Out[135…

```
0.29166666666666663
```

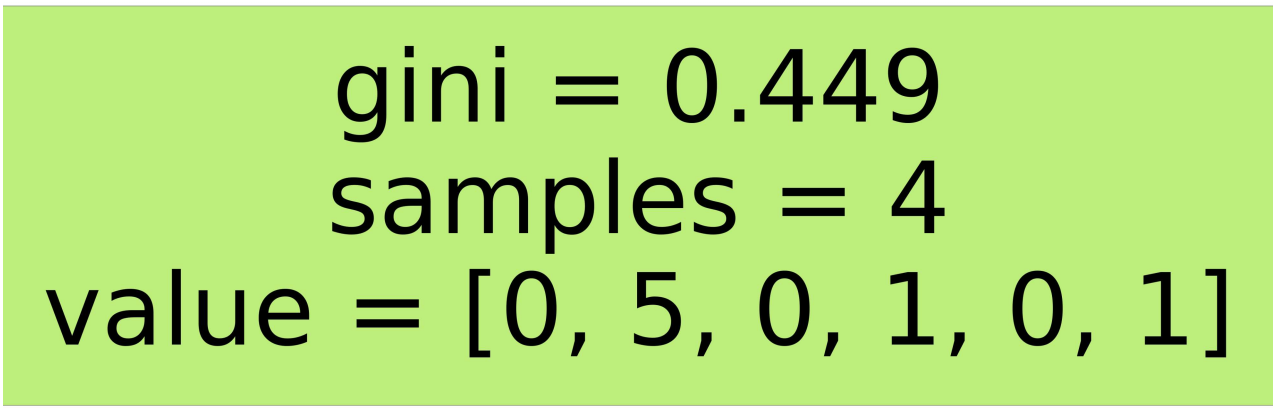
In [136…

```
rfc_best =grid_search.best_estimator_
```

In [137…

```
py.figure(figsize=(80,50))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,filled=True)
```

Out[137… [Text(2232.0, 1359.0, 'gini = 0.449\nsamples = 4\nvalue = [0, 5, 0, 1, 0, 1]')]



conclusion

The bestfit model is Logistic Regression with score of 0.9224836168176128

In [ ]:

In [ ]: