# Stations

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression,LogisticRegression,Lasso,Rid
from sklearn.model_selection import train_test_split
```

In [2]:
```
df=pd.read_csv("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 |
| **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.dropna()
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 |
| **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 |

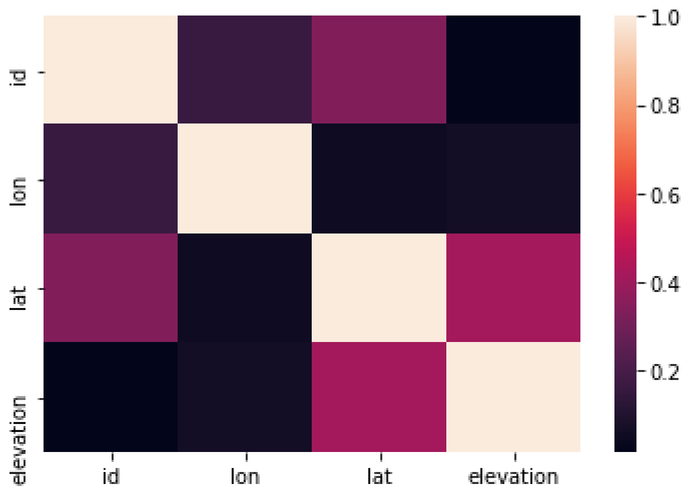
In [5]:
```
df1=df1.drop(["name","address"],axis=1)
```

In [6]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 24 entries, 0 to 23
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   id         24 non-null     int64  
 1   lon        24 non-null     float64
 2   lat        24 non-null     float64
 3   elevation  24 non-null     int64  
dtypes: float64(2), int64(2)
memory usage: 960.0 bytes
```

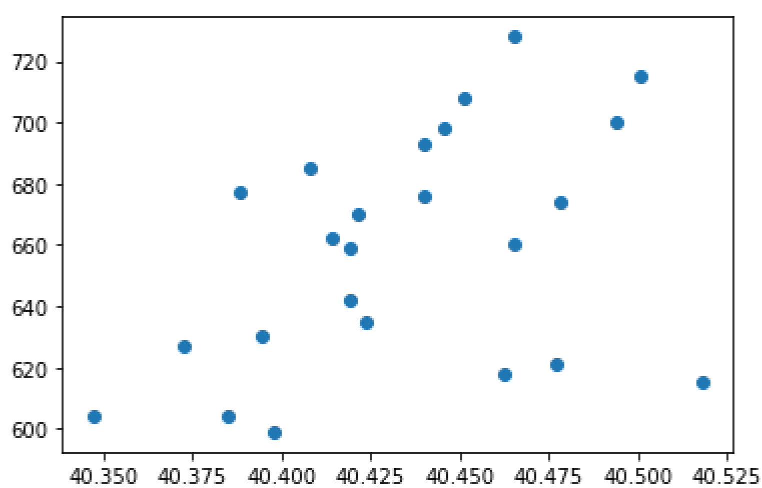
In [7]:
```
sns.heatmap(df1.corr())
```

Out[7]: <AxesSubplot:>




In [8]:
```
plt.plot(df1["lat"],df1["elevation"],"o")
```

Out[8]: [<matplotlib.lines.Line2D at 0x2a9c17eb220>]



In [9]:
```
# sns.stripplot(x=df["EBE"],y=df["PXY"],jitter=True,marker='o',color='blue')
```

In [10]:
```
x=df1.drop(["elevation"],axis=1)
y=df1["elevation"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

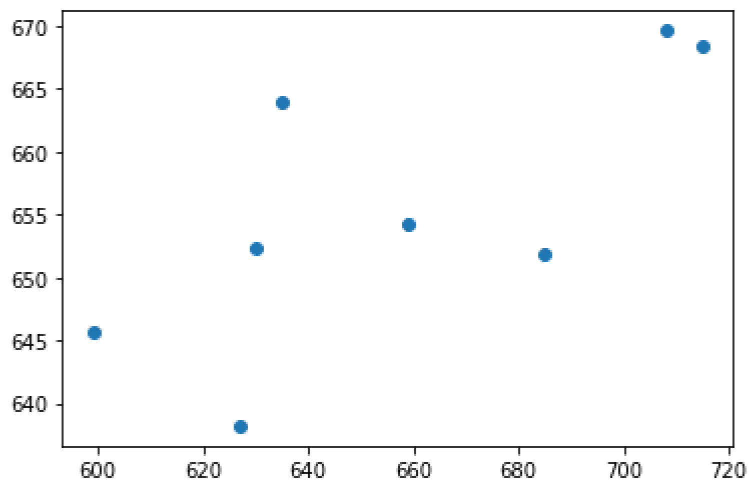
# Linear

In [11]:
```
li=LinearRegression()
li.fit(x_train,y_train)
```

Out[11]: LinearRegression()

In [12]:
```
prediction=li.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[12]: <matplotlib.collections.PathCollection at 0x2a9c1eeebb0>




In [13]:
```
lis=li.score(x_test,y_test)
```

In [14]:
```
df1["lat"].value_counts()
```

Out[14]:
```
40.423853    1
40.462531    1
40.347139    1
40.518058    1
40.419356    1
40.388153    1
40.445544    1
40.398114    1
40.414444    1
40.384964    1
40.451475    1
40.372933    1
40.465572    1
40.394781    1
40.421564    1
40.407947    1
40.478228    1
40.476928    1
40.465250    1
40.494208    1
40.439897    1
40.419208    1
40.440047    1
40.500589    1
Name: lat, dtype: int64
```

In [15]:
```
df1.loc[df1["lat"]<40.44,"lat"]=1
df1.loc[df1["lat"]>1.40,"lat"]=2
df1["lat"].value_counts()
```

Out[15]:
```
1.0    13
2.0    11
Name: lat, dtype: int64
```

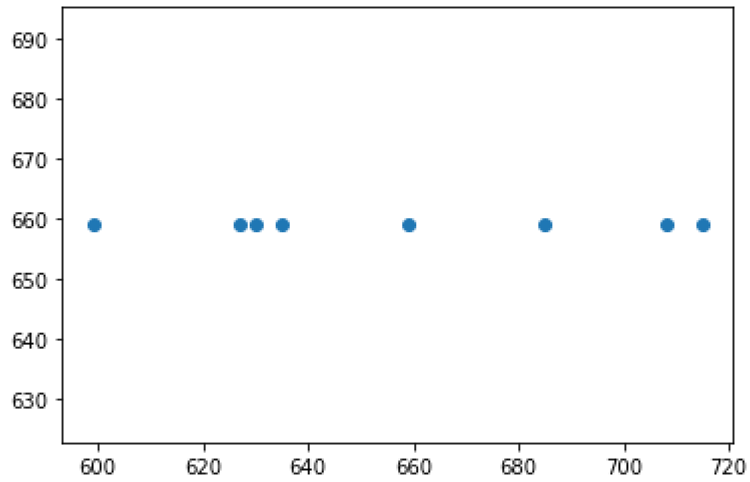
# Lasso

In [16]:
```
la=Lasso(alpha=5)
la.fit(x_train,y_train)
```

Out[16]:
```
Lasso(alpha=5)
```

In [17]:
```
prediction1=la.predict(x_test)
plt.scatter(y_test,prediction1)
```

Out[17]: <matplotlib.collections.PathCollection at 0x2a9c1f63580>




In [18]:
```
las=la.score(x_test,y_test)
```

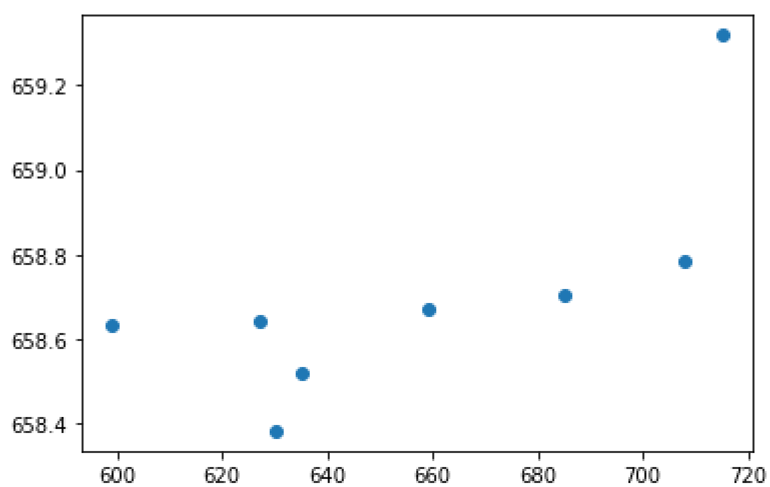
# Ridge

In [19]:
```
rr=Ridge(alpha=1)
rr.fit(x_train,y_train)
```

Out[19]: Ridge(alpha=1)

In [20]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[20]: <matplotlib.collections.PathCollection at 0x2a9c1fbac10>



In [21]:
```
rrs=rr.score(x_test,y_test)
```

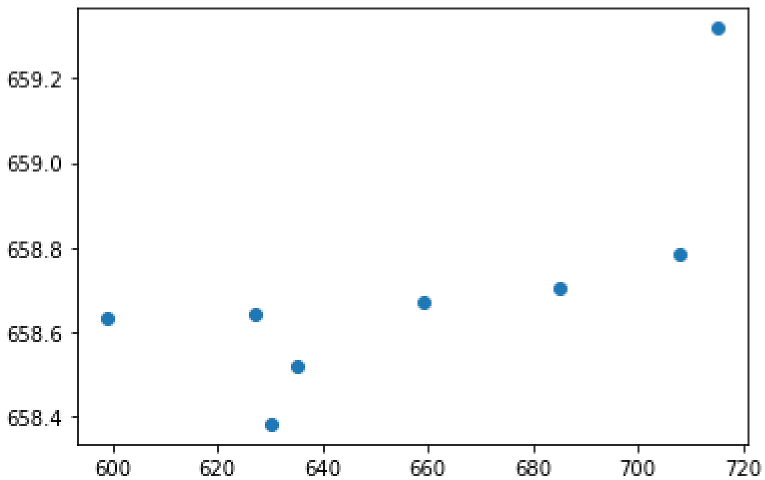
# Elastic Net

In [22]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]: ElasticNet()

In [23]:
```
prediction2=rr.predict(x_test)
plt.scatter(y_test,prediction2)
```

Out[23]: <matplotlib.collections.PathCollection at 0x2a9c1f75070>




In [24]:
```
ens=en.score(x_test,y_test)
```

# Logistic

In [25]:
```
g={"lat":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
df1["lat"].value_counts()
```

Out[25]:
```
Low     13
High    11
Name: lat, dtype: int64
```

In [26]:
```
x=df1.drop(["lat"],axis=1)
y=df1["lat"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [27]:
```
lo=LogisticRegression()
lo.fit(x_train,y_train)
```

Out[27]:
```
LogisticRegression()
```

In [28]:
```
prediction3=lo.predict(x_test)
plt.scatter(y_test,prediction3)
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x2a9c20439d0>
```


In [29]:
```
los=lo.score(x_test,y_test)
```

# Random Forest

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [31]:
```
g1={"lat":{"Low":0,"High":1}}
df1=df1.replace(g1)
```

In [32]:
```
x=df1.drop(["lat"],axis=1)
y=df1["lat"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [33]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[33]:
```
RandomForestClassifier()
```

In [34]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
}
```

In [35]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accu
grid_search.fit(x_train,y_train)
```

Out[35]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 4, 5, 6],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

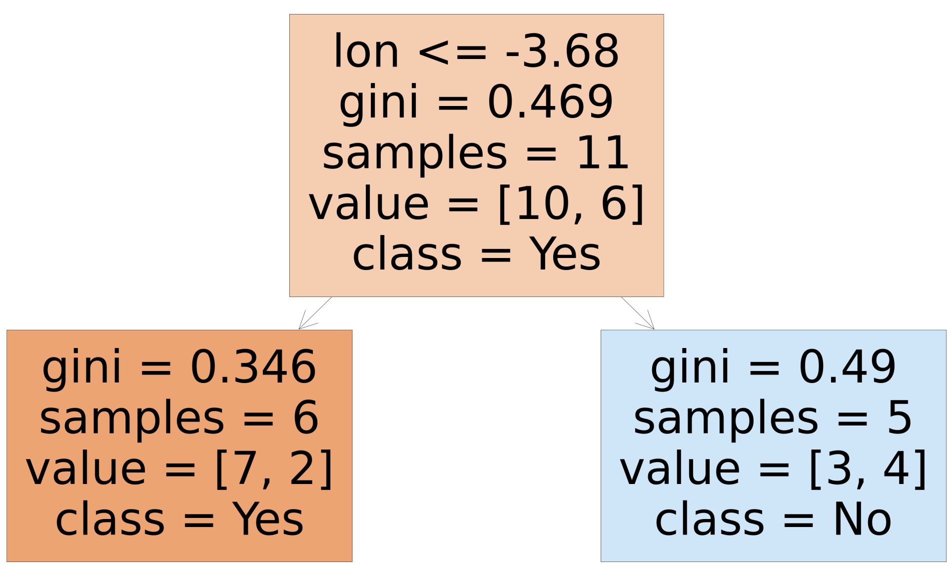
In [36]:
```
rfcs=grid_search.best_score_
```

In [37]:
```
rfc_best=grid_search.best_estimator_
```

In [38]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes',"
```

Out[38]:
```
[Text(2232.0, 1630.8000000000002, 'lon <= -3.68\ngini = 0.469\nsamples = 11\n
value = [10, 6]\nclass = Yes'),
 Text(1116.0, 543.5999999999999, 'gini = 0.346\nsamples = 6\nvalue = [7, 2]\n
class = Yes'),
 Text(3348.0, 543.5999999999999, 'gini = 0.49\nsamples = 5\nvalue = [3, 4]\nc
lass = No')]
```



In [40]:
```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
print("Random Forest:",rfcs)
```

```
Linear: 0.3130261684188642
Lasso: -0.0017273805143300791
Ridge: 0.008149999718764178
ElasticNet: -0.00137933116138389
Logistic: 0.375
Random Forest: 0.625
```

# Best Model is Random Forest