In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as pp
```

In [2]:

```
df1 = pd.read_csv(r"C:\Users\user\Desktop\c10\stations.csv")
df = df1.head(1000)
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=df.dropna()
```

In [4]:

```
df.columns
```

Out[4]:

```
Index(['id', 'name', 'address', 'lon', 'lat', 'elevation'], dtype='objec
t')
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 1.3+ KB
```

In [6]:

```
data=df[['lon' ,'lat']]
data
```

Out[6]:

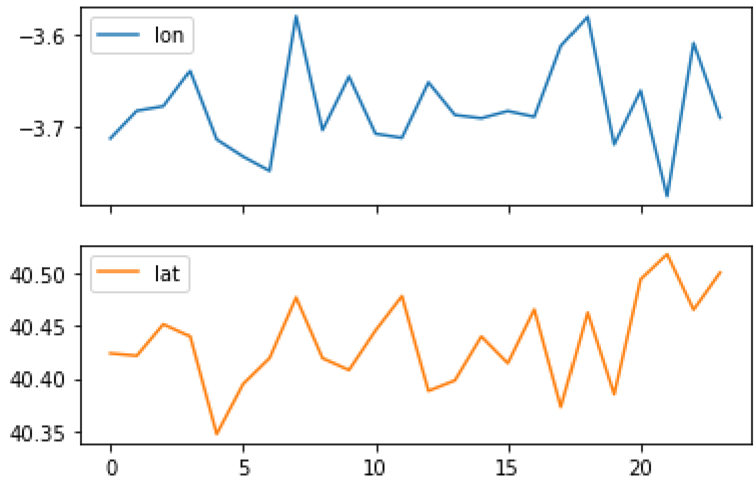
	lon	lat
0	-3.712247	40.423853
1	-3.682319	40.421564
2	-3.677356	40.451475
3	-3.639233	40.440047
4	-3.713322	40.347139
5	-3.731853	40.394781
6	-3.747347	40.419356
7	-3.580031	40.476928
8	-3.703172	40.419208
9	-3.645306	40.407947
10	-3.707128	40.445544
11	-3.711542	40.478228
12	-3.651522	40.388153
13	-3.686825	40.398114
14	-3.690367	40.439897
15	-3.682583	40.414444
16	-3.688769	40.465572
17	-3.612117	40.372933
18	-3.580747	40.462531
19	-3.718728	40.384964
20	-3.660503	40.494208
21	-3.774611	40.518058
22	-3.609072	40.465250
23	-3.689761	40.500589

In [7]:

```
data.plot.line(subplots=True)
```

Out[7]:

```
array([<AxesSubplot:>, <AxesSubplot:>], dtype=object)
```

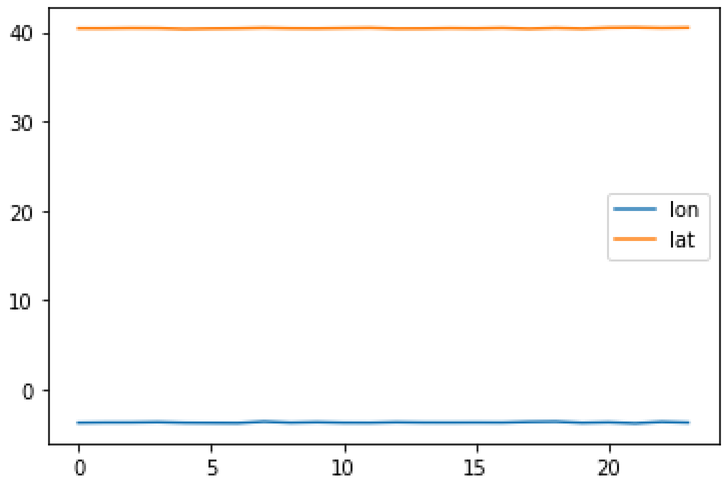


In [8]:

```
data.plot.line()
```

Out[8]:

```
<AxesSubplot:>
```



In [9]:

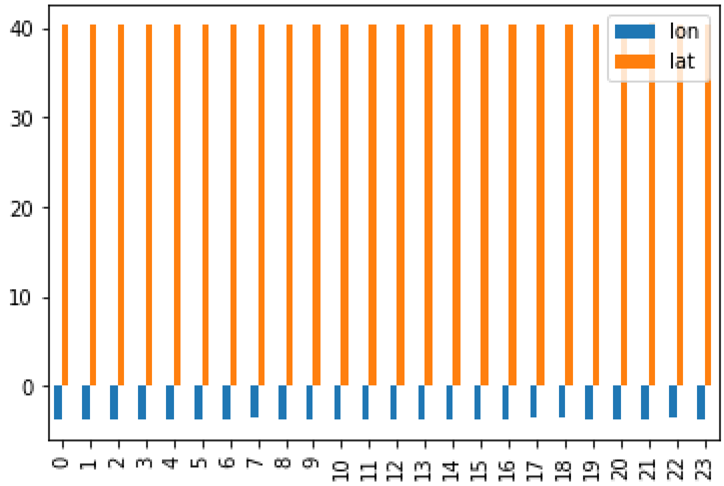
```
x = data[0:100]
```

In [10]:

```
x.plot.bar()
```

Out[10]:

<AxesSubplot:>

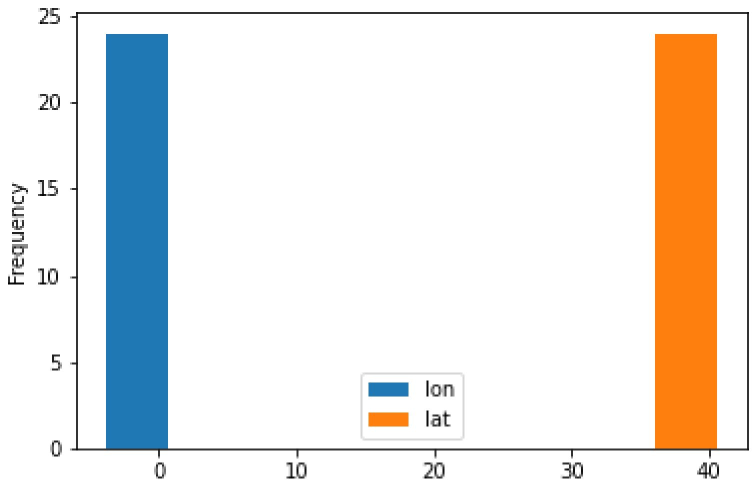


In [11]:

```
data.plot.hist()
```

Out[11]:

<AxesSubplot:ylabel='Frequency'>

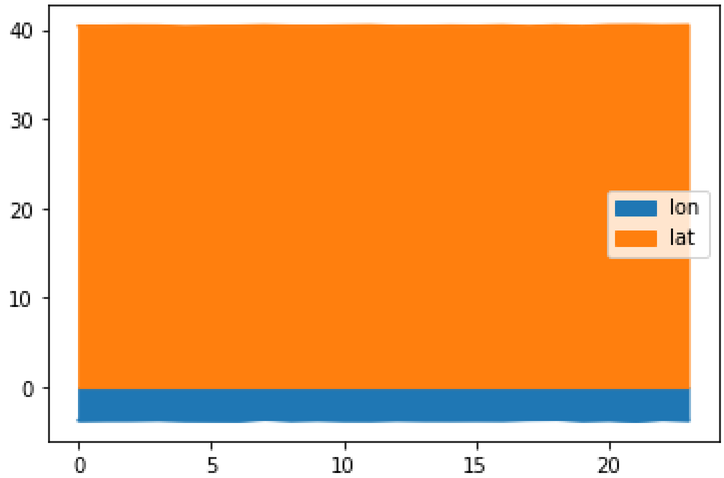


In [12]:

```
data.plot.area()
```

Out[12]:

<AxesSubplot:>

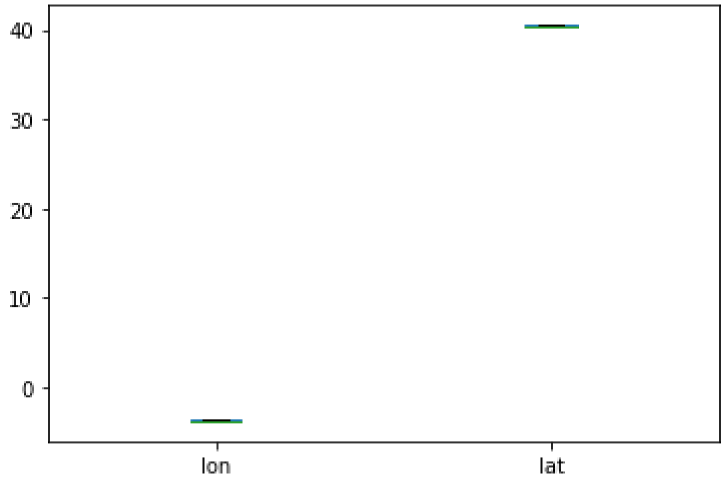


In [13]:

```
data.plot.box()
```

Out[13]:

<AxesSubplot:>

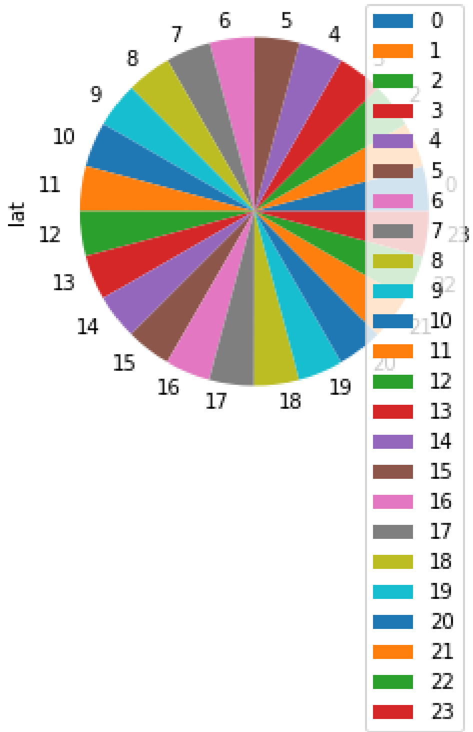


In [14]:

```
x.plot.pie(y='lat' )
```

Out[14]:

```
<AxesSubplot:ylabel='lat'>
```

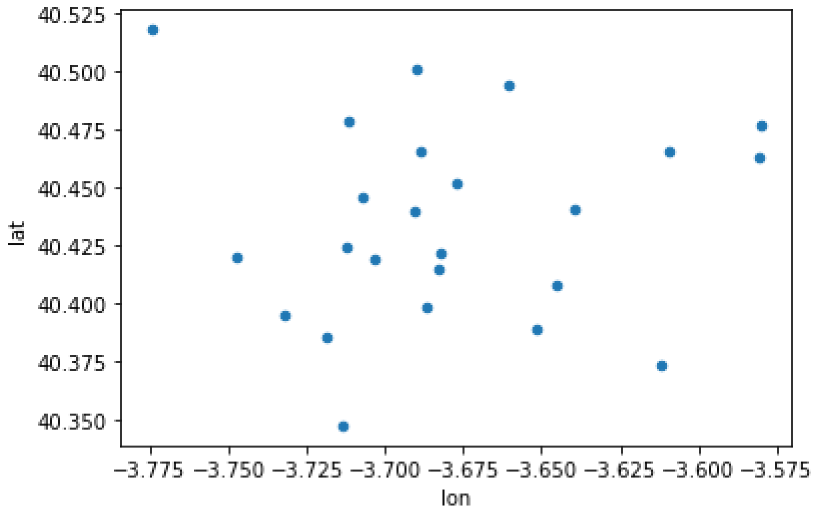


In [15]:

```
data.plot.scatter(x='lon' ,y='lat')
```

Out[15]:

```
<AxesSubplot:xlabel='lon', ylabel='lat'>
```



In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 1.3+ KB
```

In [17]:

```
df.describe()
```

Out[17]:

	id	lon	lat	elevation
count	2.400000e+01	24.000000	24.000000	24.000000
mean	2.807904e+07	-3.679019	40.434616	658.333333
std	1.799094e+01	0.049324	0.043022	38.295949
min	2.807900e+07	-3.774611	40.347139	599.000000
25%	2.807902e+07	-3.711718	40.405489	625.500000
50%	2.807904e+07	-3.687797	40.431875	661.000000
75%	2.807905e+07	-3.649968	40.465331	687.000000
max	2.807906e+07	-3.580031	40.518058	728.000000

In [18]:

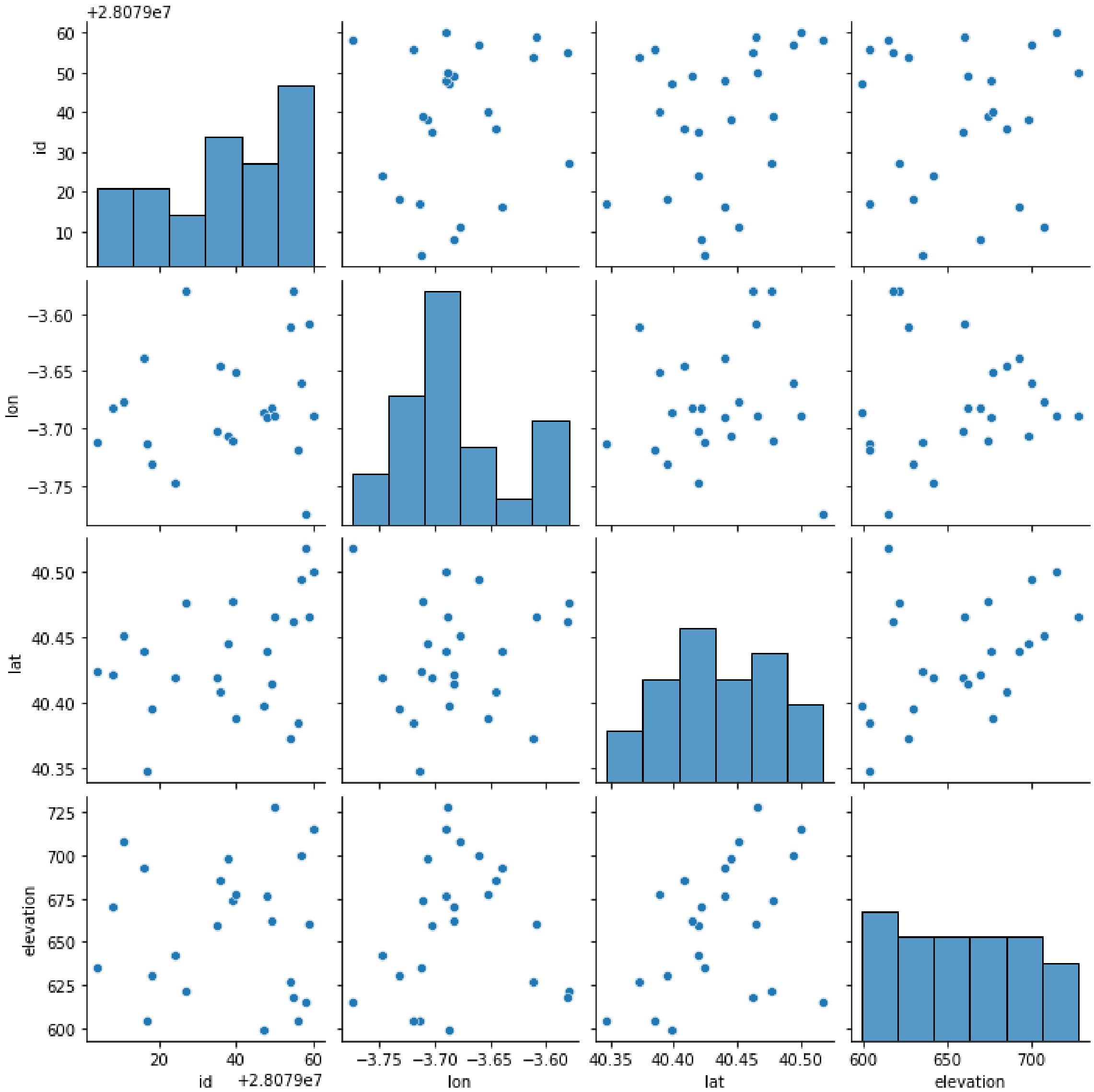
```
df1=df[['id', 'name', 'address', 'lon', 'lat', 'elevation']]
```

In [19]:

```
sb.pairplot(df1[0:100])
```

Out[19]:

```
<seaborn.axisgrid.PairGrid at 0x1f9e809f400>
```



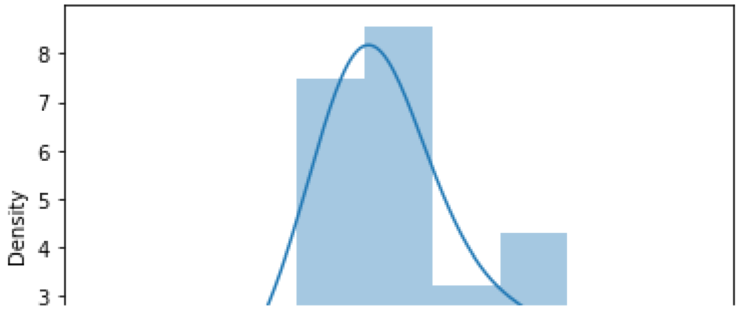
In [20]:

```
sb.distplot(df1['lon'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:255
7: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-l
evel function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]:

```
<AxesSubplot:xlabel='lon', ylabel='Density'>
```

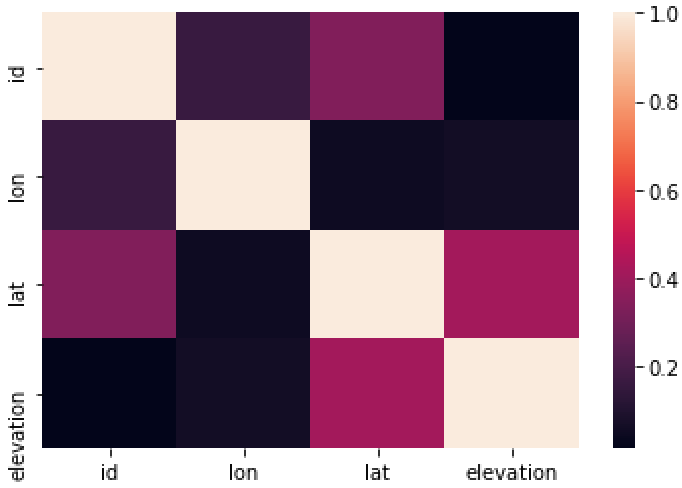


In [21]:

```
sb.heatmap(df1.corr())
```

Out[21]:

```
<AxesSubplot:>
```



In [22]:

```
x=df[['id', 'lon', 'lat', 'elevation']]
y=df['elevation']
```

In [23]:

```
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 [24]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[24]:

```
LinearRegression()
```

In [25]:

```
lr.intercept_
```

Out[25]:

```
6.227082849363796e-09
```

In [26]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[26]:

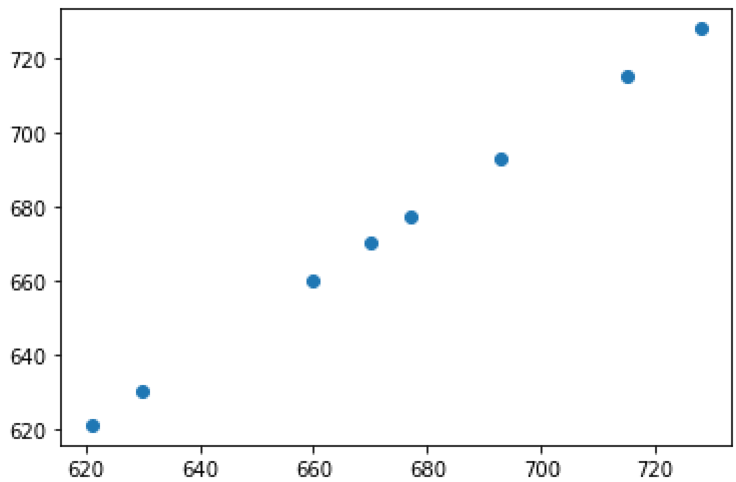
	Co-efficient
id	-2.221287e-16
lon	2.018873e-13
lat	2.522868e-13
elevation	1.000000e+00

In [27]:

```
prediction =lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[27]:

```
<matplotlib.collections.PathCollection at 0x1f9e94ea760>
```



In [28]:

```
lr.score(x_test,y_test)
```

Out[28]:

```
1.0
```

In [29]:

```
lr.score(x_train,y_train)
```

Out[29]:

```
1.0
```

In [30]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [31]:

```
r=Ridge(alpha=10)
r.fit(x_train,y_train)
```

Out[31]:

```
Ridge(alpha=10)
```

In [32]:

```
r.score(x_test,y_test)
```

Out[32]:

```
0.9999996035447566
```

In [33]:

```
r.score(x_train,y_train)
```

Out[33]:

```
0.9999997616610735
```

In [34]:

```
l=Lasso(alpha=10)
l.fit(x_train,y_train)
```

Out[34]:

```
Lasso(alpha=10)
```

In [35]:

```
l.score(x_train,y_train)
```

Out[35]:

```
0.9999406444858906
```

In [36]:

```
l.score(x_test,y_test)
```

Out[36]:

```
0.9999133689718303
```

In [37]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(x_train,y_train)
```

Out[37]:

```
ElasticNet()
```

In [38]:

```
e.coef_
```

Out[38]:

```
array([-0.        ,  0.        ,  0.        ,  0.99922987])
```

In [39]:

```
e.intercept_
```

Out[39]:

```
0.5008724113449716
```

In [40]:

```
prediction=e.predict(x_test)
```

In [41]:

```
e.score(x_test,y_test)
```

Out[41]:

```
0.9999991343567602
```

In [42]:

```
from sklearn import metrics
```

In [43]:

```
print(metrics.mean_squared_error(y_test,prediction))
```

```
0.0010737763362632603
```

In [44]:

```
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
0.0327685266111136
```

In [45]:

```
print(metrics.mean_absolute_error(y_test,prediction))
```

0.027965296078377833

In [46]:

```
from sklearn.linear_model import LogisticRegression
```

In [47]:

```
feature_matrix=df[['id']]
target_vector=df['elevation']
```

In [48]:

```
feature_matrix.shape
```

Out[48]:

(24, 1)

In [49]:

```
target_vector.shape
```

Out[49]:

(24,)

In [50]:

```
from sklearn.preprocessing import StandardScaler
```

In [51]:

```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [52]:

```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[52]:

LogisticRegression(max_iter=10000)

In [53]:

```
observation=[[1]]
```

In [54]:

```
prediction=logr.predict(observation)
print(prediction)
```

[604]

In [55]:

```
logr.classes_
```

Out[55]:

```
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 [56]:

```
logr.score(fs,target_vector)
```

Out[56]:

```
0.16666666666666666
```

In [57]:

```
logr.predict_proba(observation)[0][0]
```

Out[57]:

```
0.05149080255479361
```

In [58]:

```
logr.predict_proba(observation)
```

Out[58]:

```
array([[0.0514908 , 0.07628281, 0.06459573, 0.06111161, 0.02846647,
        0.05992926, 0.01982391, 0.00955813, 0.02542849, 0.03721458,
        0.06573175, 0.05391838, 0.01206135, 0.04186834, 0.05270507,
        0.04305319, 0.03836362, 0.0181069 , 0.04069126, 0.06344616,
        0.01416966, 0.06685293, 0.05512959]])
```

In [59]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [60]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[60]:

```
RandomForestClassifier()
```

In [64]:

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

In [65]:

```
from sklearn.model_selection import GridSearchCV
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: UserWarning: The least populated class in y has only 1 members, wh
ich is less than n_splits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[65]:

```
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 [66]:

```
rfc_best=grid_search.best_estimator_
```

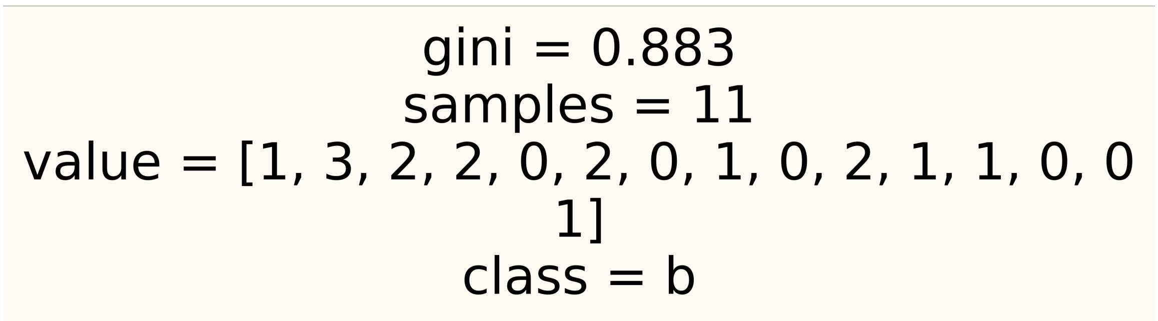
In [67]:

```
from sklearn.tree import plot_tree

pp.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d'],f
```

Out[67]:

```
[Text(2232.0, 1087.2, 'gini = 0.883\nsamples = 11\nvalue = [1, 3, 2, 2, 0,
2, 0, 1, 0, 2, 1, 1, 0, 0\n1]\nclass = b')]
```



logistic regression is suitable (0.16666666666666666)

In [ ]:

```

```