Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing Datasets

In [2]:

```
df=pd.read_csv("stations1")
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

Data Cleaning and Data Preprocessing

In [3]:
```
df=df.dropna()
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['id', 'name', 'address', 'lon', 'lat', 'elevation'], dtype='object')
```

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]:

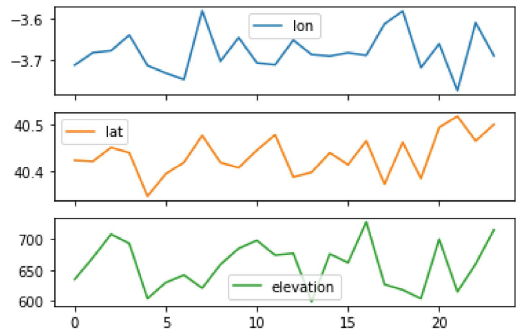
```
data=df[['lon', 'lat', 'elevation']]
data
```

Out[6]:

	lon	lat	elevation
0	-3.712247	40.423853	635
1	-3.682319	40.421564	670
2	-3.677356	40.451475	708
3	-3.639233	40.440047	693
4	-3.713322	40.347139	604
5	-3.731853	40.394781	630
6	-3.747347	40.419356	642
7	-3.580031	40.476928	621
8	-3.703172	40.419208	659
9	-3.645306	40.407947	685
10	-3.707128	40.445544	698
11	-3.711542	40.478228	674
12	-3.651522	40.388153	677
13	-3.686825	40.398114	599
14	-3.690367	40.439897	676
15	-3.682583	40.414444	662
16	-3.688769	40.465572	728
17	-3.612117	40.372933	627
18	-3.580747	40.462531	618
19	-3.718728	40.384964	604
20	-3.660503	40.494208	700
21	-3.774611	40.518058	615
22	-3.609072	40.465250	660
23	-3.689761	40.500589	715

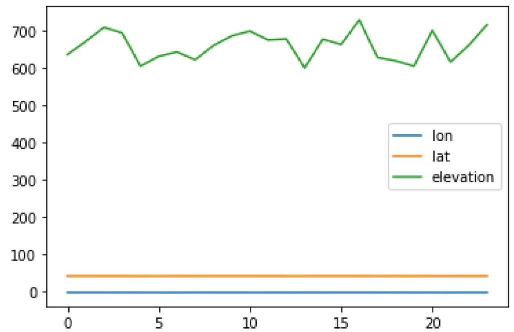
Line chart

In [7]:
```
data.plot.line(subplots=True)
```
Out[7]: array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>], dtype=object)



Line chart

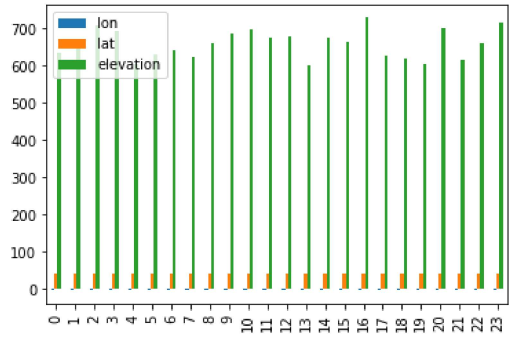
In [8]:
```
data.plot.line()
```
Out[8]: <AxesSubplot:>



Bar chart

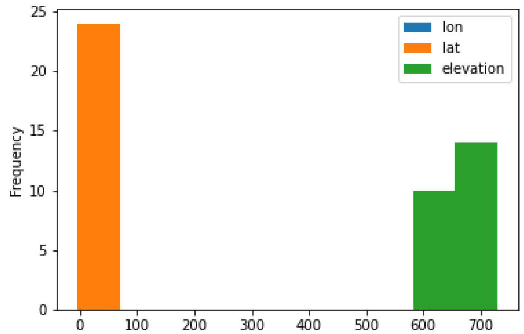
In [9]:
```
b=data[0:50]
```

In [10]:
```
b.plot.bar()
```
Out[10]: <AxesSubplot:>



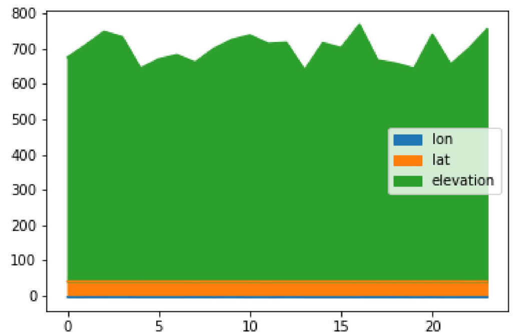
Histogram

In [11]:
```
data.plot.hist()
```
Out[11]: <AxesSubplot:ylabel='Frequency'>



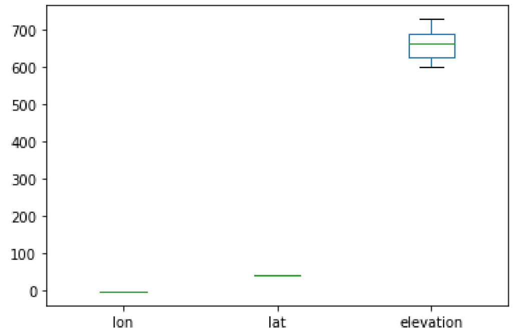
Area chart

In [12]:
```
data.plot.area()
```
Out[12]: <AxesSubplot:>



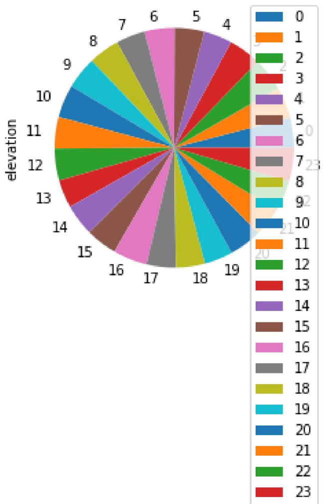
Box chart

In [13]:
```
data.plot.box()
```
Out[13]: <AxesSubplot:>



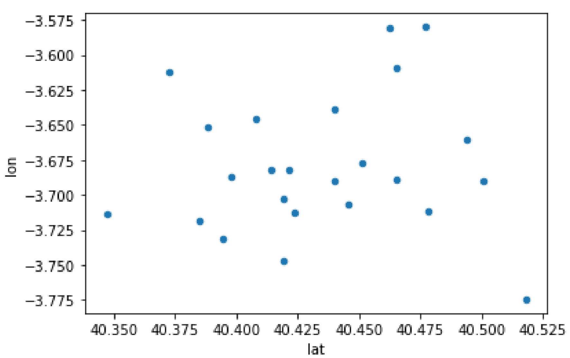
Pie chart

In [14]:
```
b.plot.pie(y= 'elevation' )
```
Out[14]: <AxesSubplot:ylabel='elevation'>



Scatter chart

In [15]:
```
data.plot.scatter(x='lat' ,y='lon')
```
Out[15]: <AxesSubplot:xlabel='lat', ylabel='lon'>



In [16]:
```
1 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]:
```
1 df.describe()
```

Out[17]:

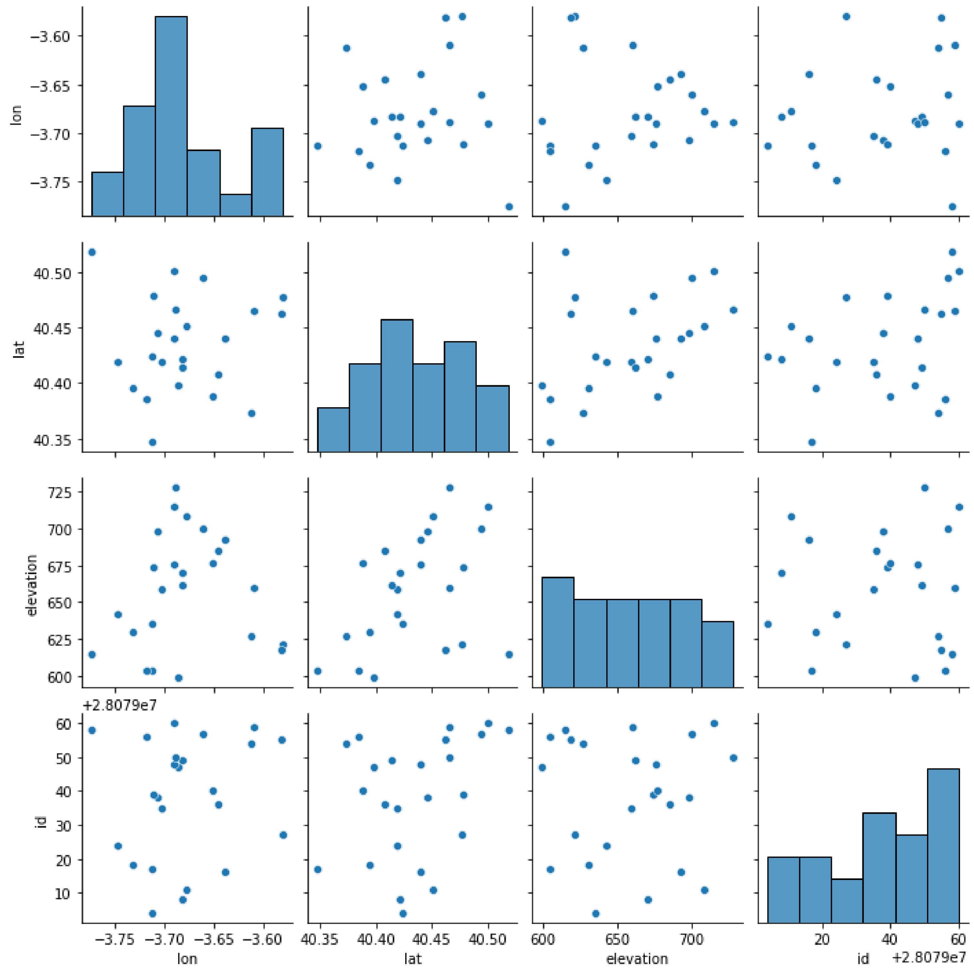
	id	lon	lat	elevation
count	2.400000e+01	24.000000	24.000000	24.000000
mean	2.807904e+07	-3.679019	40.434616	658.333333
std	1.799094e+01	0.049324	0.043022	38.295949
min	2.807900e+07	-3.774611	40.347139	599.000000
25%	2.807902e+07	-3.711718	40.405489	625.500000
50%	2.807904e+07	-3.687797	40.431875	661.000000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

In [18]:
```
1 df1=df[['lon', 'lat', 'elevation','id']]
```

EDA AND VISUALIZATION

In [19]:
```
sns.pairplot(df1[0:50])
```

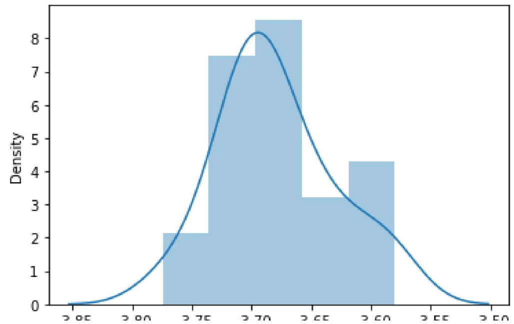
Out[19]: <seaborn.axisgrid.PairGrid at 0x18019d7d1f0>



In [20]:
```
sns.distplot(df1['lon'])
```

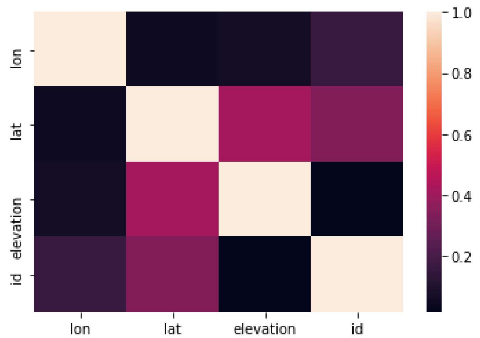
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function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

Out[20]: <AxesSubplot:xlabel='lon', ylabel='Density'>



In [21]:
```
sns.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [22]:
```
x=df[['id']]
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)
```

Linear Regression

In [24]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

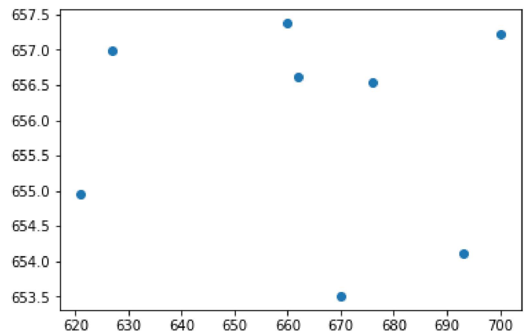
Out[24]: LinearRegression()

In [25]:
```
lr.intercept_
```
Out[25]: -2129342.4097207235

In [26]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```
Out[26]:

	Co-efficient
id	0.075857

In [27]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```
Out[27]: <matplotlib.collections.PathCollection at 0x1801b37ca00>



ACCURACY

In [28]:
```
lr.score(x_test,y_test)
```
Out[28]: -0.09486822413919249

In [29]:
```
lr.score(x_train,y_train)
```
Out[29]: 0.0009583837933925254

Ridge and Lasso

In [30]:
```
from sklearn.linear_model import Ridge,Lasso
```
In [31]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```
Out[31]: Ridge(alpha=10)

Accuracy(Ridge)

In [32]:
```
rr.score(x_test,y_test)
```
Out[32]: -0.09485306914926017

In [33]:
```
rr.score(x_train,y_train)
```
Out[33]: 0.0009583793573364474

In [34]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[34]: Lasso(alpha=10)

In [35]:
```
la.score(x_train,y_train)
```
Out[35]: 0.0007600890445066399

Accuracy(Lasso)

In [36]:
```
la.score(x_test,y_test)
```
Out[36]: -0.09226892618842597

In [37]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[37]: ElasticNet()

In [38]:
```
en.coef_
```
Out[38]: array([0.07400431])

In [39]:
```
en.intercept_
```
Out[39]: -2077313.9320505918

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

In [41]:
```
en.score(x_test,y_test)
```
Out[41]: -0.09469779583781057

Evaluation Metrics

In [42]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```
```
23.697995156078832
759.97684010734
27.5676774521783
```

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
feature_matrix=df[['lon', 'lat', 'elevation']]
target_vector=df[ 'id']
```

In [45]:
```
feature_matrix.shape
```

Out[45]:
```
(24, 3)
```

In [46]:
```
target_vector.shape
```

Out[46]:
```
(24,)
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
```

In [48]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [49]:
```
logr=LogisticRegression(max_iter=10000)
logr.fit(fs,target_vector)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
observation=[[1,2,3]]
```

In [51]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[28079060]
```

In [52]:
```
logr.classes_
```

Out[52]:
```
array([28079004, 28079008, 28079011, 28079016, 28079017, 28079018,
       28079024, 28079027, 28079035, 28079036, 28079038, 28079039,
       28079040, 28079047, 28079048, 28079049, 28079050, 28079054,
       28079055, 28079056, 28079057, 28079058, 28079059, 28079060],
      dtype=int64)
```

In [53]:
```
logr.score(fs,target_vector)
```

Out[53]:
```
0.625
```

In [54]:
```
logr.predict_proba(observation)[0][0]
```

Out[54]:
```
0.002670094237l3907
```

In [55]:
```
logr.predict_proba(observation)
```

Out[55]:
```
array([[2.67009424e-03, 1.31718443e-02, 9.28886937e-02, 7.05390759e-02,
        8.34704249e-05, 8.61239187e-04, 1.92424615e-03, 9.42530382e-03,
        6.74552285e-03, 2.41930772e-02, 3.99013815e-02, 3.17626697e-02,
        1.03584997e-02, 3.64133186e-04, 2.15799332e-02, 8.44889971e-03,
        1.94987201e-01, 1.02771134e-03, 5.74880974e-03, 2.41257564e-04,
        1.88161280e-01, 2.77774487e-03, 4.15162302e-02, 2.30621680e-01]])
```

Random Forest

In [56]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [57]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[57]:
```
RandomForestClassifier()
```

In [58]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]
}
```

In [59]:
```
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, which is less than n_sp
lits=2.
  warnings.warn(("The least populated class in y has only %d"
```

Out[59]:
```
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 [60]:
```
grid_search.best_score_
```

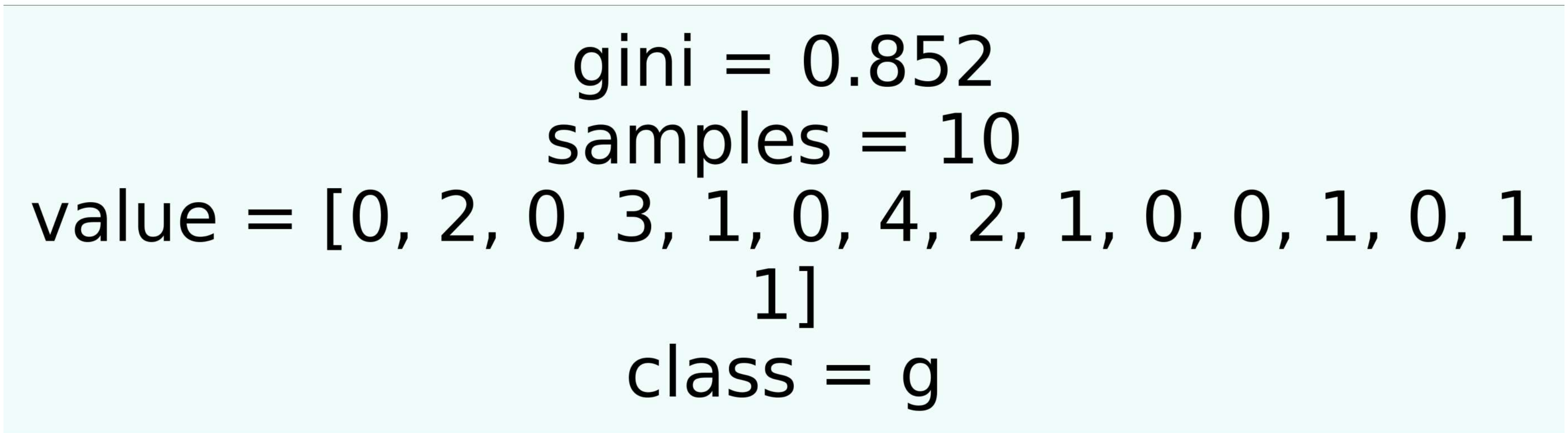
Out[60]:
```
0.125
```

In [61]:
```
rfc_best=grid_search.best_estimator_
```

In [66]:
```
rn.tree import plot_tree

(figsize=(80,40))
rfc_best.estimators_[5],feature_names=x.columns,class_names=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x'],filled=True)
```

Out[66]:
```
[Text(2232.0, 1087.2, 'gini = 0.852\nsamples = 10\nvalue = [0, 2, 0, 3, 1, 0, 4, 2, 1, 0, 0, 1, 0, 1\n1]\nclass = g')]
```



Conclusion

Accuracy

Linear Regression:0.0009583837933925254

Ridge Regression:0.0009583793573364474 ¶

Lasso Regression:0.09226892618842597

ElasticNet Regression:0.09469779583781057

Logistic Regression:0.625

Random Forest:0.125

Logistic Regression is suitable for this dataset