Analysis and Prediction of Coal Mines all over India.

India is the world's second largest producer of coal after China, having cumulative total coal resources of 319.020 billion tonnes(2k18) and is dependent on coal for many of its core sectors.

Coal is the largest source of electricity generation in India and as per reports, fuels approximately 74% of India's electricity.

In addition to power and electricity, sectors like iron and steel, cement and other industries like fertilisers, pulp and paper are also among the largest consumers of coal in India.

Coal has been mined in India for about two centuries.
Developments in the post-independence period have been significant and strides made since nationalization of the coal industry have been even more impressive.

Coal will continue to be India's prime source of energy for power generation, steel-making, powering of locomotives, and production of cement, fertilizer and domestic fuel.

In this Small Project of mine, I am trying to Analyse the data from the production of coal in the country. And will try to do the prediction using Machine Learning Algorithms !

In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

Importing the Datasets

Production of Coal Throughout the country in year 2019-2020

In [26]:
```
df = pd.read_csv('Coal Mines Dataset india.csv')
```

In [27]:
```
df.head()
```

Out[27]:

	State/UT Name	Mine Name	Coal/ Lignite Production	Coal Mine Owner Name	Coal/Lignite	Govt Owned/Private	Type of Mine	Latitude	Longit
0	West Bengal	Ningah Colliery	0.01	ECL	Coal	G	UG	23.6743	87.0
1	West Bengal	Jhanjhara Project Colly	3.50	ECL	Coal	G	UG	23.6680	87.2
2	West Bengal	MAOHUSUDANPUR	0.04	ECL	Coal	G	UG	23.6338	87.2
3	West Bengal	PARASCOLE(EAST)	0.10	ECL	Coal	G	UG	23.6377	87.1
4	West Bengal	PARASCOLE(WEST)	0.07	ECL	Coal	G	UG	23.6405	87.1

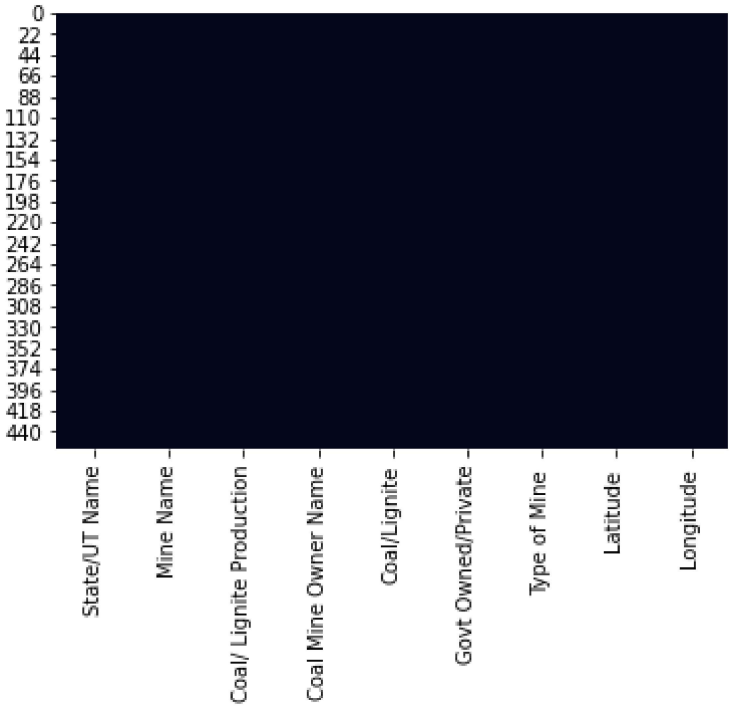
In [28]:
```
df.tail()
```

Out[28]:

	State/UT Name	Mine Name	Coal/ Lignite Production	Coal Mine Owner Name	Coal/Lignite	Govt Owned/Private	Type of Mine	Latitude	Longitude
454	Telangana	GDK 1&3 INC	0.214798	SCCL	Coal	SG	UG	18.0452	79.3025
455	Telangana	GDK 2&2A INC	0.283656	SCCL	Coal	SG	UG	18.4221	79.3111
456	Telangana	GDK 5 INC	0.018202	SCCL	Coal	SG	UG	18.4545	79.3546
457	Telangana	GDK 11 INC	0.804029	SCCL	Coal	SG	UG	18.4161	79.3258
458	Telangana	GDK 10 INC	0.019551	SCCL	Coal	SG	UG	18.3915	79.3345

In [29]:
```
# Checking for null values
sns.heatmap(df.isnull(), cbar=False)
```

Out[29]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x172f322c4e0>
```



So, we can see, this dataset does not have any null values !

Analysis and Visualization -

All CoalMine Owner in India According to Dataset

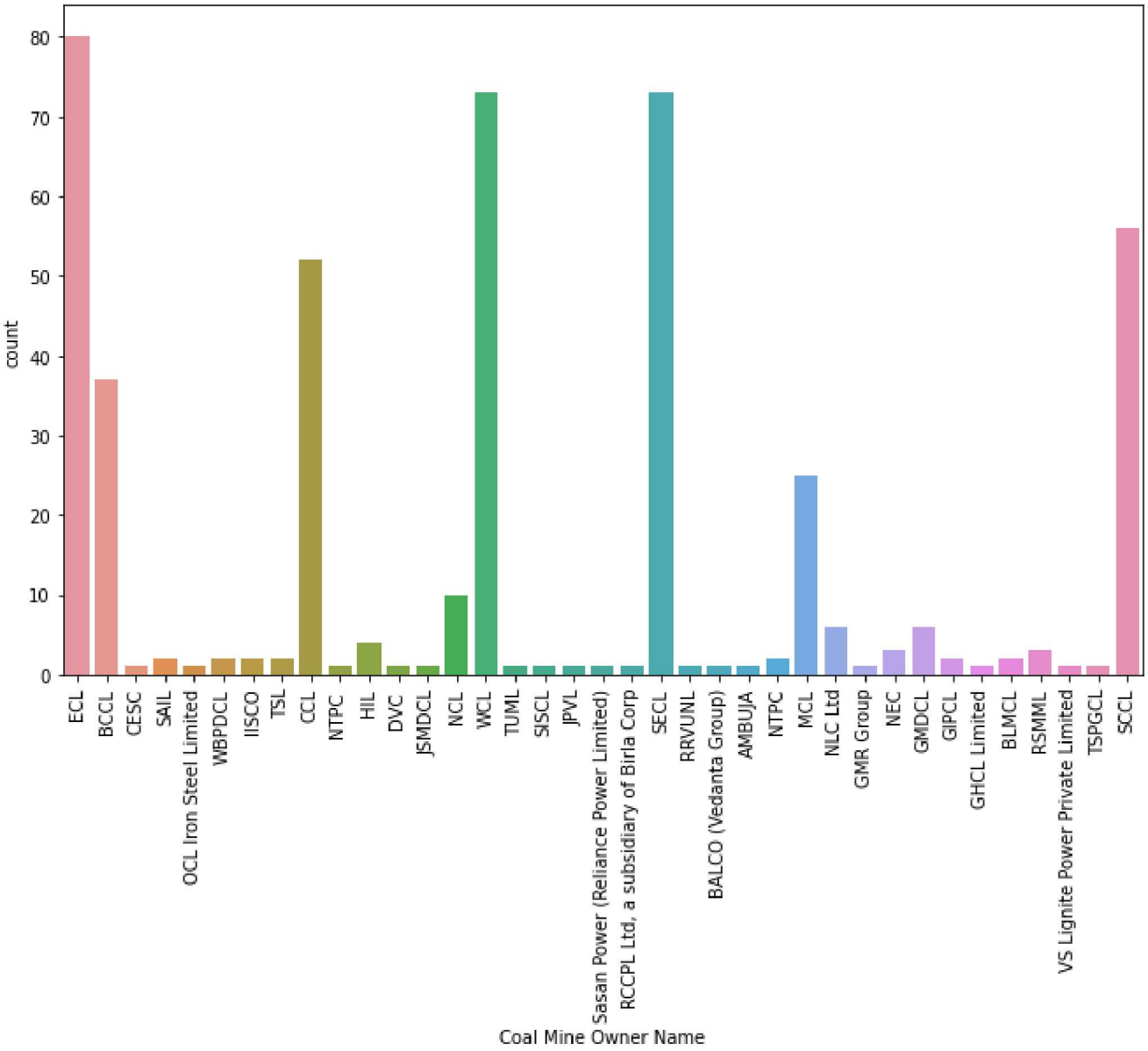
In [30]:
```
arr = df["Coal Mine Owner Name"].unique()
```

In [31]:
```
arr
```

Out[31]:
```
array(['ECL', 'BCCL', 'CESC', 'SAIL', 'OCL Iron Steel Limited', 'WBPDCL',
       'IISCO', 'TSL', 'CCL', 'NTPC ', 'HIL', 'DVC ', 'JSMDCL', 'NCL',
       'WCL', 'TUML', 'SISCL', 'JPVL',
       'Sasan Power (Reliance Power Limited)',
       'RCCPL Ltd, a subsidiary of Birla Corp', 'SECL', 'RRVUNL',
       'BALCO (Vedanta Group)', 'AMBUJA', 'NTPC', 'MCL', 'NLC Ltd',
       'GMR Group', 'NEC', 'GMDCL', 'GIPCL', 'GHCL Limited', 'BLMCL',
       'RSMML', 'VS Lignite Power Private Limited', 'TSPGCL', 'SCCL'],
      dtype=object)
```

In [52]:
```
plt.figure(figsize=(11,7))
g= sns.countplot("Coal Mine Owner Name", data=df )
g.set_xticklabels(g.get_xticklabels(), rotation=90)
```

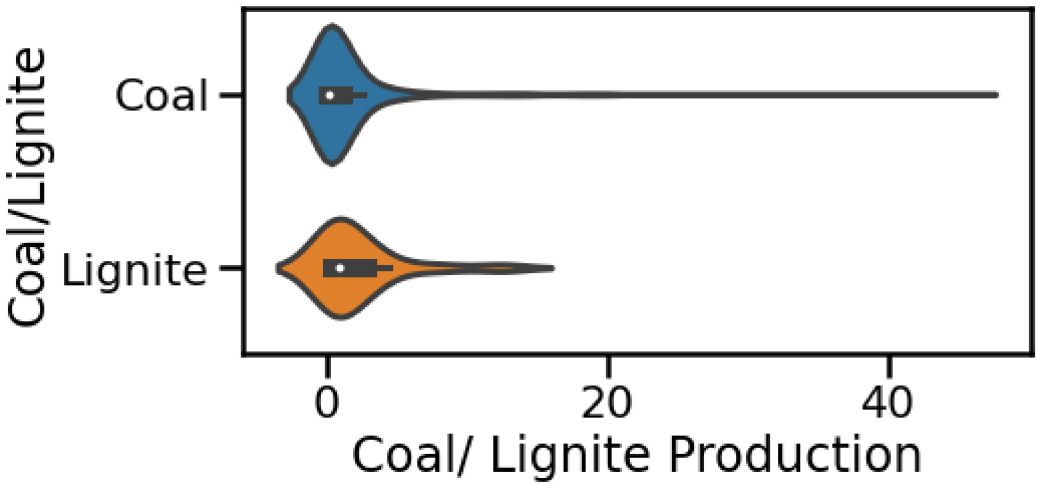
Out[52]: [Text(0, 0, 'ECL'),
 Text(0, 0, 'BCCL'),
 Text(0, 0, 'CESC'),
 Text(0, 0, 'SAIL'),
 Text(0, 0, 'OCL Iron Steel Limited'),
 Text(0, 0, 'WBPDCL'),
 Text(0, 0, 'IISCO'),
 Text(0, 0, 'TSL'),
 Text(0, 0, 'CCL'),
 Text(0, 0, 'NTPC '),
 Text(0, 0, 'HIL'),
 Text(0, 0, 'DVC '),
 Text(0, 0, 'JSMDCL'),
 Text(0, 0, 'NCL'),
 Text(0, 0, 'WCL'),
 Text(0, 0, 'TUML'),
 Text(0, 0, 'SISCL'),
 Text(0, 0, 'JPVL'),
 Text(0, 0, 'Sasan Power (Reliance Power Limited)'),
 Text(0, 0, 'RCCPL Ltd, a subsidiary of Birla Corp'),
 Text(0, 0, 'SECL'),
 Text(0, 0, 'RRVUNL'),
 Text(0, 0, 'BALCO (Vedanta Group)'),
 Text(0, 0, 'AMBUJA'),
 Text(0, 0, 'NTPC'),
 Text(0, 0, 'MCL'),
 Text(0, 0, 'NLC Ltd'),
 Text(0, 0, 'GMR Group'),
 Text(0, 0, 'NEC'),
 Text(0, 0, 'GMDCL'),
 Text(0, 0, 'GIPCL'),
 Text(0, 0, 'GHCL Limited'),
 Text(0, 0, 'BLMCL'),
 Text(0, 0, 'RSMML'),
 Text(0, 0, 'VS Lignite Power Private Limited'),
 Text(0, 0, 'TSPGCL'),
 Text(0, 0, 'SCCL')]



Coal and Lignite Produced

In [150…

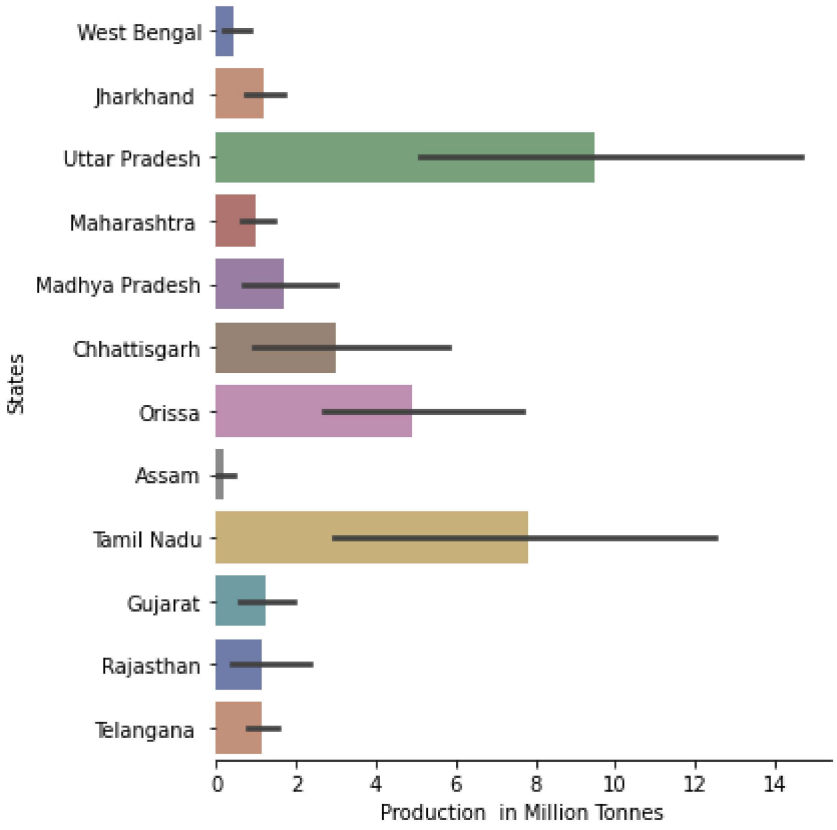
```
plt.figure(figsize=(8,4))
sns.set_context('poster')

sns.violinplot(y="Coal/Lignite", x = "Coal/ Lignite Production", data=df,
               inplace =True, inner ="box", )
plt.tight_layout()
```



Coal Production in Different States

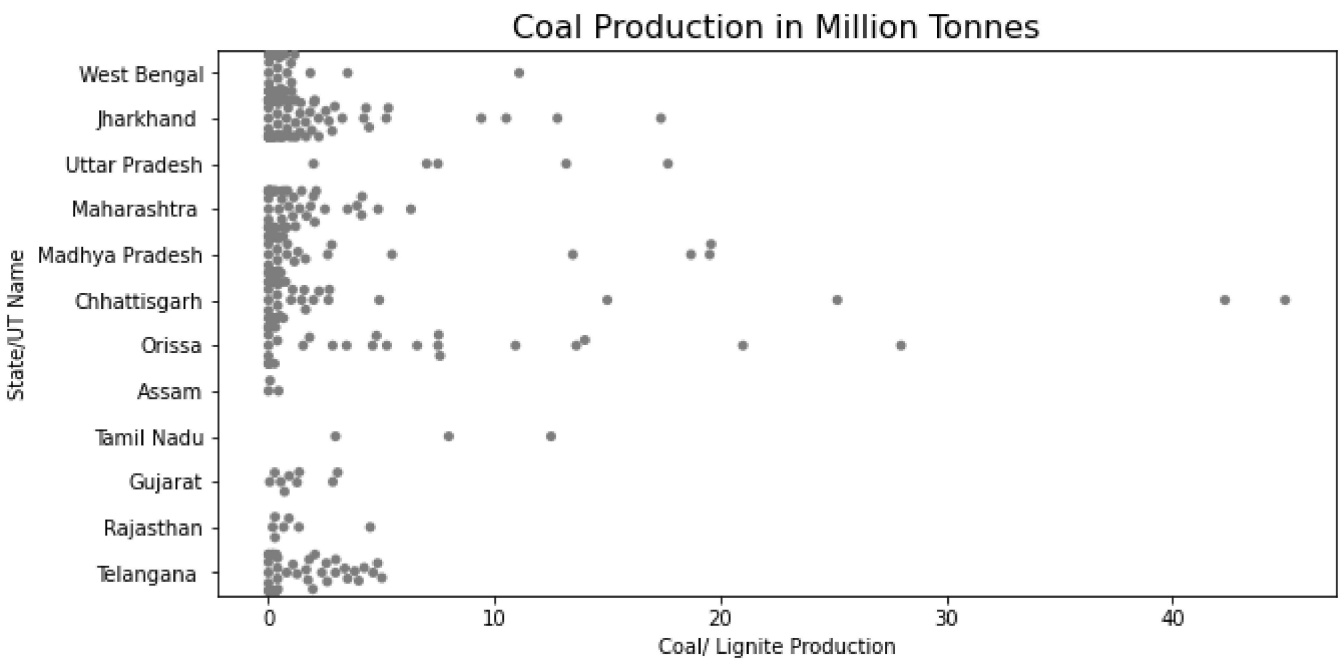
In [51]:
```
g = sns.catplot(
    data=df, kind="bar",
    x="Coal/ Lignite Production", y="State/UT Name", palette="dark", alpha=.6, height=6
)
g.despine(left=True)
g.set_axis_labels("Production  in Million Tonnes", "States")
```

Out[51]:
```
<seaborn.axisgrid.FacetGrid at 0x172f5345358>
```



In [79]:
```
plt.figure(figsize=(10,5))

sns.swarmplot(y = df['State/UT Name'], x = df['Coal/ Lignite Production'], color = "grey")
plt.title("Coal Production in Million Tonnes", size=16)
```

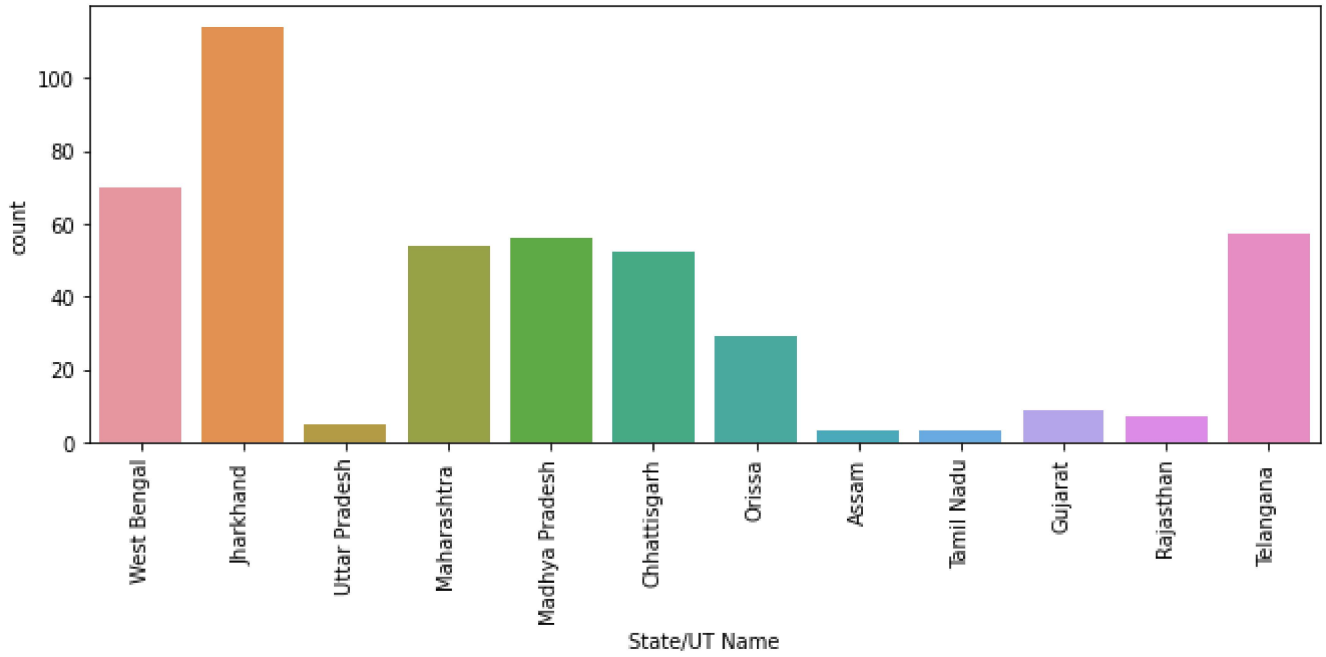
Out[79]:
```
Text(0.5, 1.0, 'Coal Production in Million Tonnes')
```



And , no. of Coalmines per state

In [54]:
```
plt.figure(figsize=(11,4))
g= sns.countplot("State/UT Name", data=df )
g.set_xticklabels(g.get_xticklabels(), rotation=90)
```

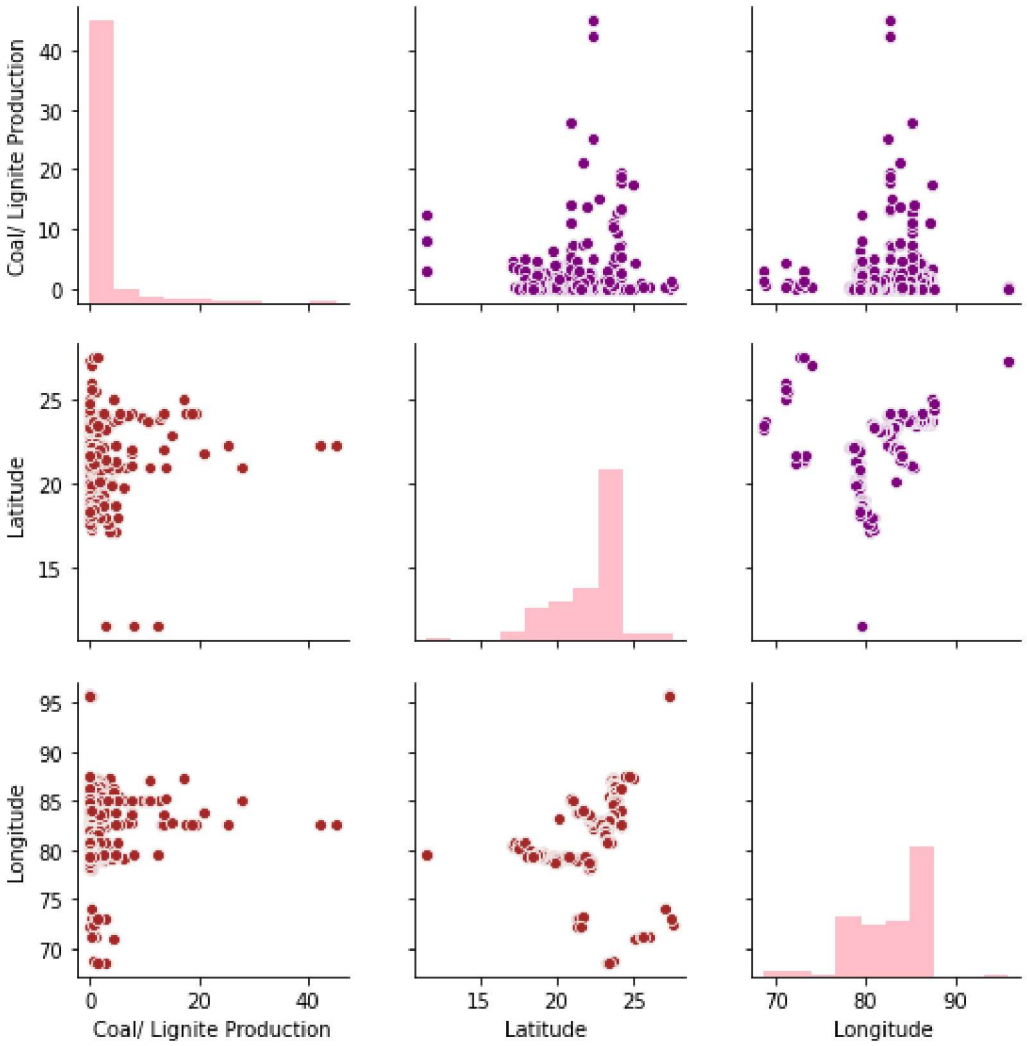
Out[54]:
```
[Text(0, 0, 'West Bengal'),
 Text(0, 0, 'Jharkhand '),
 Text(0, 0, 'Uttar Pradesh'),
 Text(0, 0, 'Maharashtra '),
 Text(0, 0, 'Madhya Pradesh'),
 Text(0, 0, 'Chhattisgarh'),
 Text(0, 0, 'Orissa'),
 Text(0, 0, 'Assam'),
 Text(0, 0, 'Tamil Nadu'),
 Text(0, 0, 'Gujarat'),
 Text(0, 0, 'Rajasthan'),
 Text(0, 0, 'Telangana ')]
```



Plotting a Scatterplot

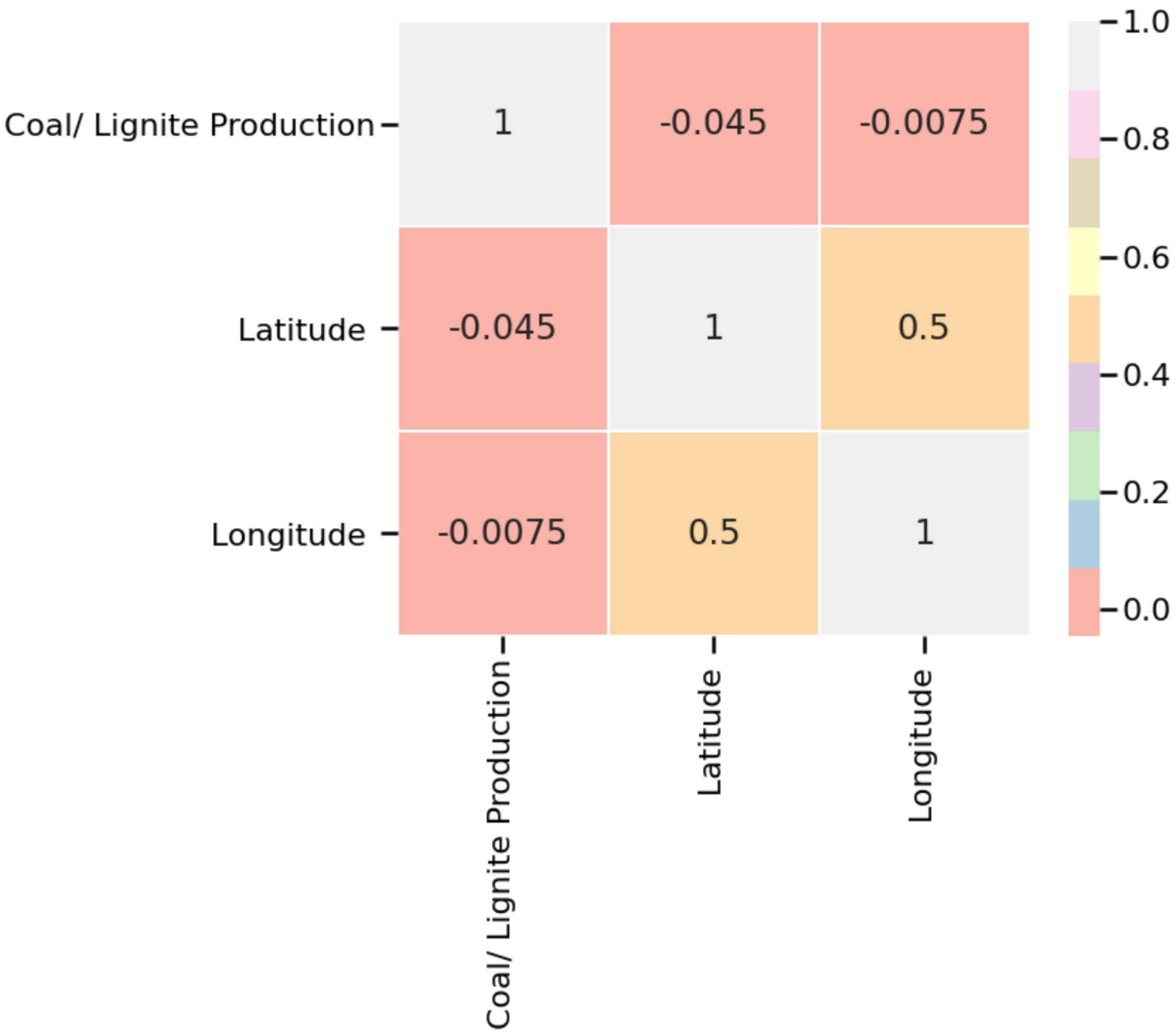
In [64]:
```
g = sns.PairGrid(df)
g.map_upper(sns.scatterplot,color='purple')
g.map_lower(sns.scatterplot, color='brown')
g.map_diag(plt.hist, color = 'pink')
```

Out[64]: <seaborn.axisgrid.PairGrid at 0x172f7e7e7b8>



And a Heatmap of features

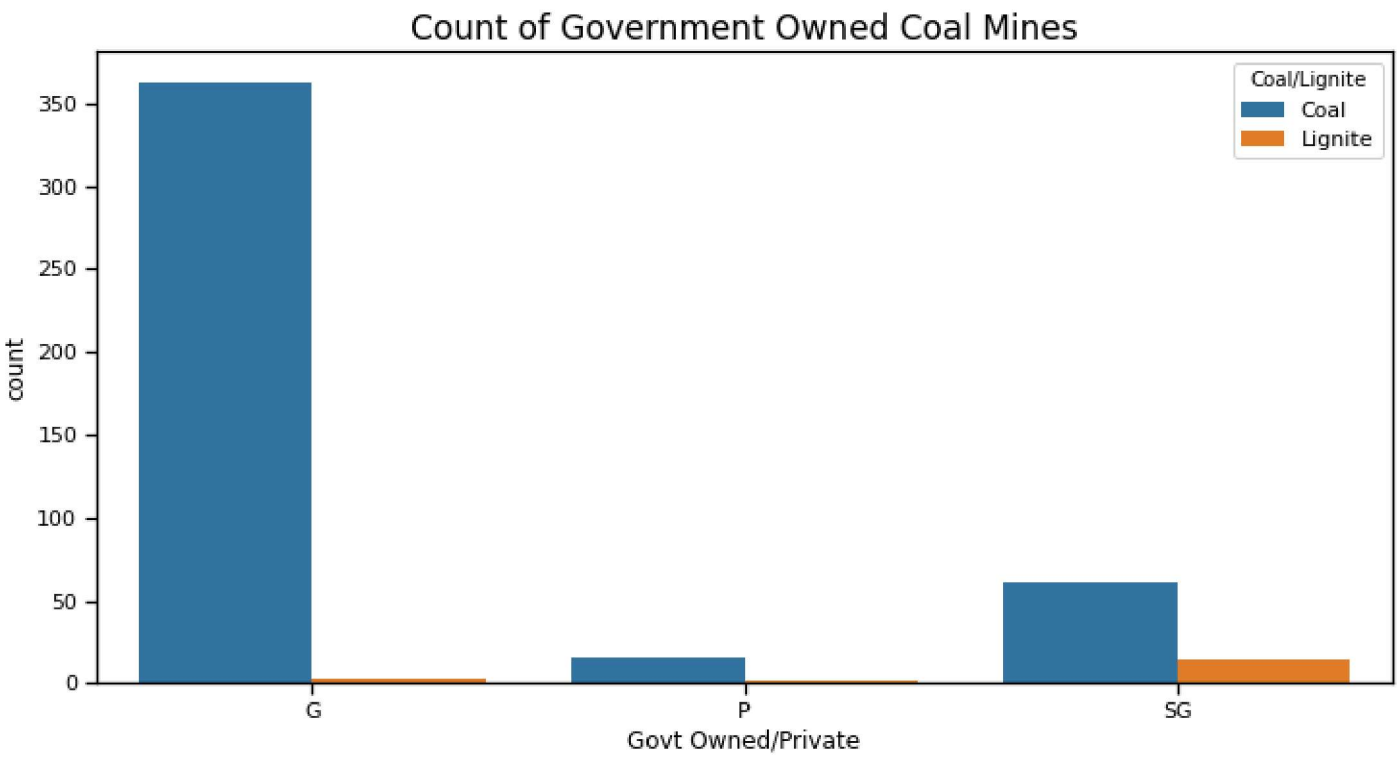
In [113…
```
corrmat = df.corr()
corr_features = corrmat.index
plt.figure(figsize=(10,8))

g=sns.heatmap(df[corr_features].corr(),annot=True, linewidth =.9 , cmap="Pastel1")
```



Plotting for Mines owned by Government

In [130…

```
plt.figure(figsize=(12,6))
plt.title("Count of Government Owned Coal Mines", size = 17)
g= sns.countplot("Govt Owned/Private", data=df , hue ='Coal/Lignite')
```



G- Government Owned

P- Privately Owned

SG - Semi Government Owned

Taking a Look at Another dataset

Statewise Production in a year (Production in Million Tonnes of COAL)

In [98]:
```
df2 = pd.read_csv('Coal Reserves State wise million tonnes.csv')
```

In [99]:
```
df2.head()
```

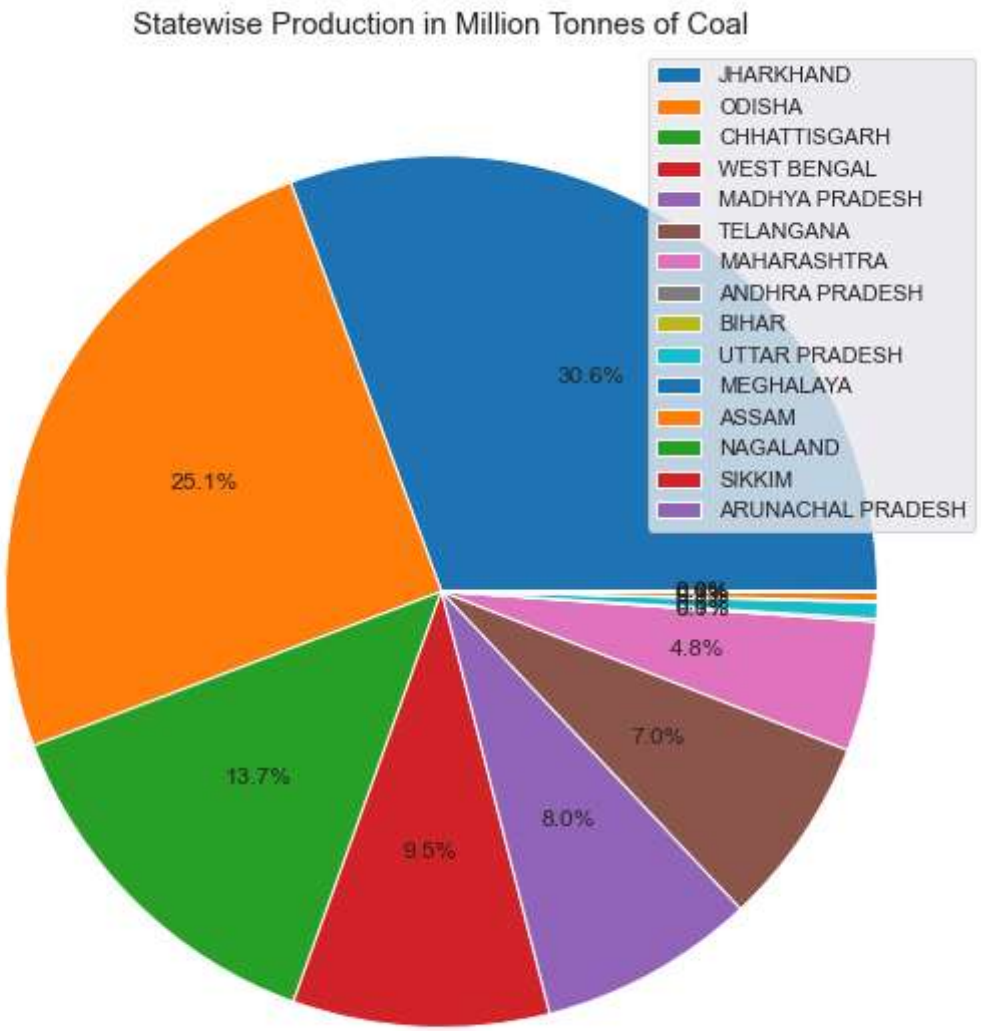
Out[99]:

	State	Proved	Indicated	Inferred
0	JHARKHAND	45563	31439	6150
1	ODISHA	37391	34165	7739
2	CHHATTISGARH	20428	34576	2202
3	WEST BENGAL	14156	12869	4643
4	MADHYA PRADESH	11958	12154	3875

Production of Coal in a year Proved per state

In [212…
```
plt.figure(figsize=(10,10))
plt.title("Statewise Production in Million Tonnes of Coal",fontsize=15)
sns.set_context('notebook')
```

```
plt.pie(df2["Proved"],autopct="%1.1f%%")
plt.legend(df2['State'],loc='best')
plt.show()
```



Showing the above column in a stacked graph

In [126…
```
df2_new = df2.set_index('State', drop=True)
```

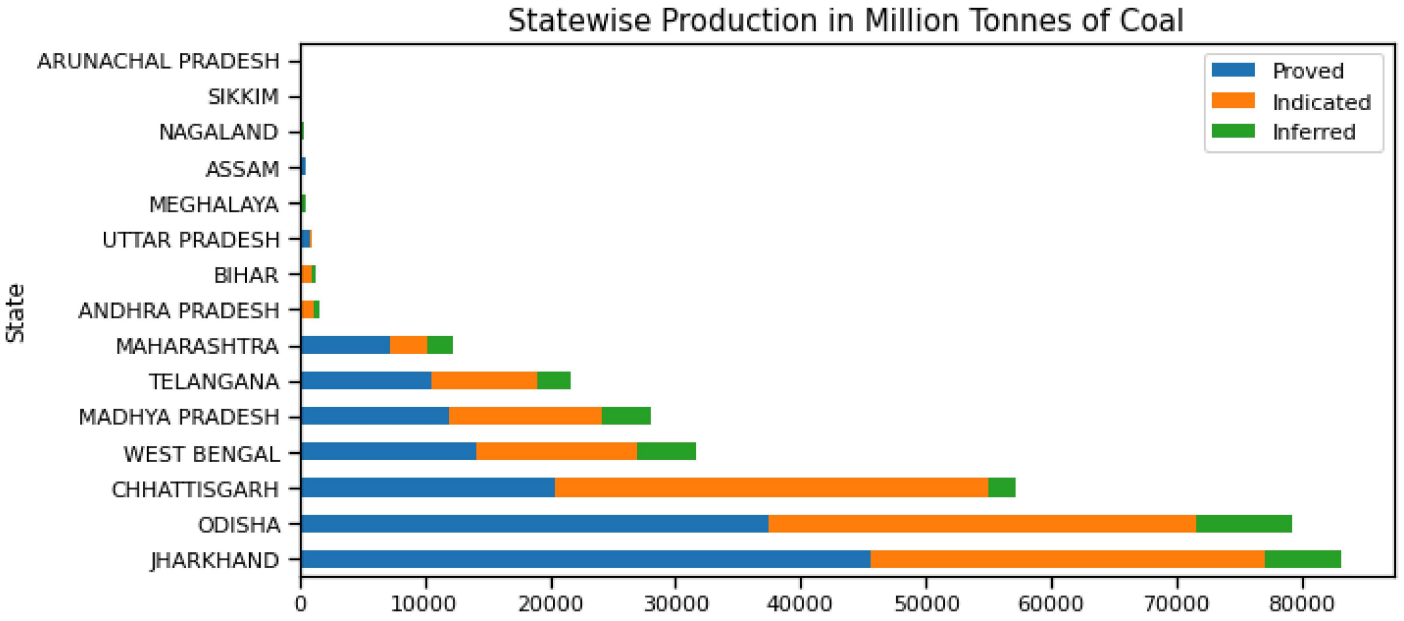
In [127…
```
df2_new.head()
```

Out[127…

State	Proved	Indicated	Inferred
JHARKHAND	45563	31439	6150
ODISHA	37391	34165	7739
CHHATTISGARH	20428	34576	2202
WEST BENGAL	14156	12869	4643
MADHYA PRADESH	11958	12154	3875

In [135…
```
df2_new.plot.barh(stacked=True,figsize=(10,5))
plt.title("Statewise Production in Million Tonnes of Coal",fontsize=15)
```

Out[135… Text(0.5, 1.0, 'Statewise Production in Million Tonnes of Coal')



Now, For Prediction...

In [148…
```
# Dropping Type of Mine
df.drop(['Type of Mine'],axis=1, inplace=True)
```

In [145…
```
df.head()
```

Out[145…

	State/UT Name	Mine Name	Coal/ Lignite Production	Coal Mine Owner Name	Coal/Lignite	Govt Owned/Private	Latitude	Longitude
0	West Bengal	Ningah Colliery	0.01	ECL	Coal	G	23.6743	87.0333
1	West Bengal	Jhanjhara Project Colly	3.50	ECL	Coal	G	23.6680	87.2963
2	West Bengal	MAOHUSUDANPUR	0.04	ECL	Coal	G	23.6338	87.2037
3	West Bengal	PARASCOLE(EAST)	0.10	ECL	Coal	G	23.6377	87.1951
4	West Bengal	PARASCOLE(WEST)	0.07	ECL	Coal	G	23.6405	87.1855

Reordering Columns

In [151…
```
title = list(df.columns)
title
```

Out[151…
```
['State/UT Name',
 'Mine Name',
 'Coal/ Lignite Production',
 'Coal Mine Owner Name',
 'Coal/Lignite',
 'Govt Owned/Private',
 'Latitude ',
 'Longitude ']
```

In [152…
```
title[0],title[2] = title[2],title[0]
title
```

Out[152…
```
['Coal/ Lignite Production',
 'Mine Name',
 'State/UT Name',
 'Coal Mine Owner Name',
 'Coal/Lignite',
 'Govt Owned/Private',
 'Latitude ',
 'Longitude ']
```

In [153…
```
df=df[title]
```

In [154…
```
df.head()
```

Out[154…

	Coal/ Lignite Production	Mine Name	State/UT Name	Coal Mine Owner Name	Coal/Lignite	Govt Owned/Private	Latitude	Longitude
0	0.01	Ningah Colliery	West Bengal	ECL	Coal	G	23.6743	87.0333
1	3.50	Jhanjhara Project Colly	West Bengal	ECL	Coal	G	23.6680	87.2963
2	0.04	MAOHUSUDANPUR	West Bengal	ECL	Coal	G	23.6338	87.2037
3	0.10	PARASCOLE(EAST)	West Bengal	ECL	Coal	G	23.6377	87.1951
4	0.07	PARASCOLE(WEST)	West Bengal	ECL	Coal	G	23.6405	87.1855

One hot Encoding - Getting Dummy Variables

In [155…
```
df=pd.get_dummies(df,drop_first=True)
```

In [156…
```
df.head()
```

Out[156…

	Coal/ Lignite Production	Latitude	Longitude	Mine Name_A A D OCM	Mine Name_ADASA UG TO OC	Mine Name_AGKCC	Mine Name_AKK OCP	Mine Name_ALP	Nam
0	0.01	23.6743	87.0333	0	0	0	0	0	
1	3.50	23.6680	87.2963	0	0	0	0	0	
2	0.04	23.6338	87.2037	0	0	0	0	0	
3	0.10	23.6377	87.1951	0	0	0	0	0	
4	0.07	23.6405	87.1855	0	0	0	0	0	

5 rows × 511 columns

Independent and Dependent Features

In [158…
```
# Production Column being the Dependent Feature..
X=df.iloc[:,1:]  #independent
y=df.iloc[:,0]
```

In [159…
```
y.head()
```

Out[159…
```
0    0.01
1    3.50
2    0.04
3    0.10
4    0.07
Name: Coal/ Lignite Production, dtype: float64
```

Extra trees Regressor Model

In [160…
```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)
```

Out[160…
```
ExtraTreesRegressor()
```

In [162…
```
#print(model.feature_importances_)
```

Train-Test Split

In [164…
```
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, random_state=0)
```

Using Random Forest Regressor

In [165…
```
from sklearn.ensemble import RandomForestRegressor
```

Hyperparameter Tuning

In [166…
```
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

Using RandomizdSearchCV for hyperparameters

In [167…
```
from sklearn.model_selection import RandomizedSearchCV
```

In [168…
```
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]

max_features = ['auto', 'sqrt']

max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]

min_samples_split = [2, 5, 10, 15, 100]
min_samples_leaf = [1, 2, 5, 10]
```

In [169…
```
## Selecting Parameters

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features':
['auto', 'sqrt'], 'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100],
'min_samples_leaf': [1, 2, 5, 10]}
```

In [170…
```
rf = RandomForestRegressor()
```

In [171…
```
# Applying randomizedsearchcv

rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_me
```

Fitting the Model

In [172…
```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   1.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   0.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   0.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   0.7s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   0.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators
=300; total time=   0.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators
=300; total time=   0.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators
=300; total time=   0.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators
=300; total time=   0.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators
=300; total time=   0.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=4
00; total time=   0.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=4
00; total time=   0.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=4
00; total time=   0.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=4
00; total time=   0.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=4
00; total time=   0.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=
700; total time=   1.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=
700; total time=   1.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=
700; total time=   1.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=
700; total time=   1.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=
700; total time=   1.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1
000; total time=   1.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1
000; total time=   1.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1
000; total time=   1.2s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1
000; total time=   1.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1
000; total time=   1.3s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=
```

```
1100; total time=   0.9s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=
1100; total time=   1.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=
1100; total time=   0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=
300; total time=   0.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=
300; total time=   0.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=
300; total time=   0.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=
300; total time=   0.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=
300; total time=   0.2s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=7
00; total time=   0.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=7
00; total time=   0.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=7
00; total time=   0.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=7
00; total time=   0.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=7
00; total time=   0.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=
700; total time=   2.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=
700; total time=   2.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=
700; total time=   2.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=
700; total time=   2.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=
700; total time=   2.4s
```

Out[172…

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000, 1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

Looking for best parameters

In [173…

```
rf_random.best_params_
```

Out[173…

```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 25}
```

In [174…
```
rf_random.best_score_
```

Out[174…
```
-13.610615603273326
```

Doing The Prediction of The Amount of Coal

In [175…
```
predict=rf_random.predict(X_test)
```

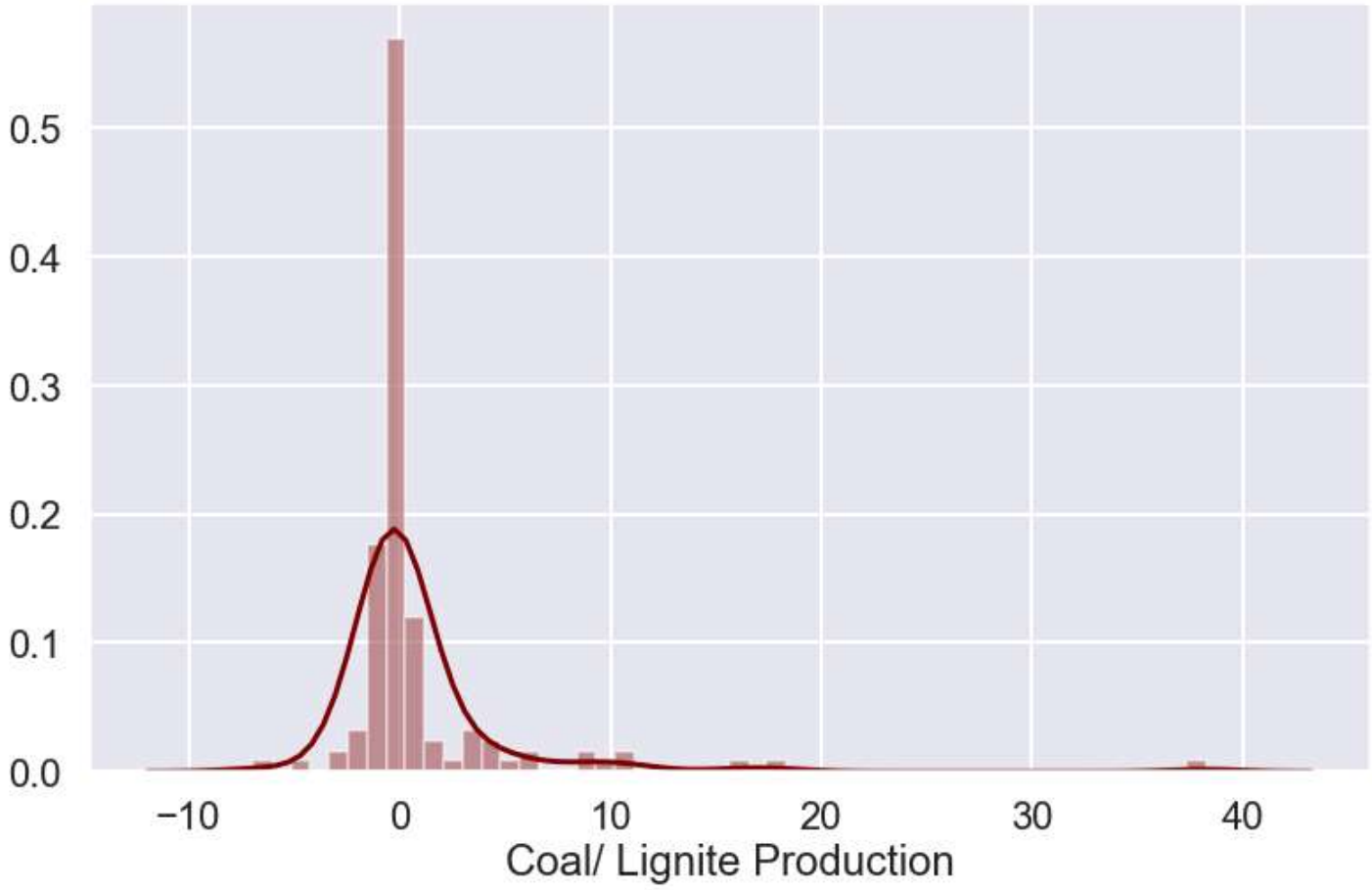
In [176…
```
predict
```

Out[176…
```
array([0.66651444, 0.49166996, 6.73959105, 0.55815262, 0.6522384 ,
       0.31971461, 1.7851105 , 0.22351245, 0.58862056, 0.64712075,
       0.85662729, 0.62177011, 0.21843506, 0.76973187, 3.93235406,
       0.56040426, 0.40945425, 0.73984518, 0.56014268, 0.45449011,
       0.22317227, 0.2863116 , 0.57917403, 0.91259427, 0.22860793,
       0.92005072, 1.72260029, 1.98178725, 0.40378476, 1.84813654,
       0.66254045, 0.86476662, 0.6910071 , 0.8866686 , 2.590328  ,
       0.73260893, 0.79604307, 0.44961299, 0.84883195, 0.18873083,
       0.85286725, 0.95572948, 0.24306979, 0.59230866, 0.42172678,
       0.73007845, 8.52825817, 1.44885578, 3.07539893, 0.54412637,
       9.75418595, 0.45439741, 1.17899645, 0.83286301, 0.62132191,
       0.60534153, 0.80840325, 1.58680199, 0.26622168, 1.8651461 ,
       1.95536489, 0.65541304, 0.26350803, 0.26660992, 0.72636589,
       0.75832636, 4.65878752, 0.78065731, 0.73411791, 0.86629391,
       0.78841937, 2.72406076, 3.89793942, 0.80036146, 4.55396298,
       0.80500437, 0.67315161, 0.20162206, 5.19940463, 0.65543115,
       0.77993044, 0.68257072, 0.52606599, 0.88244218, 0.27603309,
       0.68301739, 0.73301721, 0.31468428, 1.39982596, 0.69798044,
       0.42673678, 0.7216257 , 0.67203851, 3.55673417, 0.22499522,
       0.27099427, 0.65403703, 0.21417264, 0.31095558, 2.01341194,
       3.47468525, 0.20630015, 0.31468428, 0.86260384, 0.75609596,
       0.85643113, 0.22470829, 1.11580715, 0.21951606, 0.26136945,
       0.31468428, 0.6506141 , 0.26348501, 7.66457363, 1.15456382,
       0.22623447, 0.19775249, 0.46405002, 0.22397083, 1.2730068 ,
       0.99501518, 0.21568074, 1.39411928, 0.31468428, 0.86457804,
       8.9711807 , 0.65318118, 0.70089642, 0.49907178, 0.82190482,
       0.76766263, 0.40040476, 0.44867475, 3.07416253, 1.56518108,
       0.7303664 , 0.77548326, 0.81629036])
```

Now, Plotting the Prediction :

In [192…
```
plt.figure(figsize=(13,8))
sns.distplot(y_test-predict, color ="maroon")
```

Out[192…
```
<matplotlib.axes._subplots.AxesSubplot at 0x172fd073e80>
```



YAY!! We can see a Normal Distribution Curve

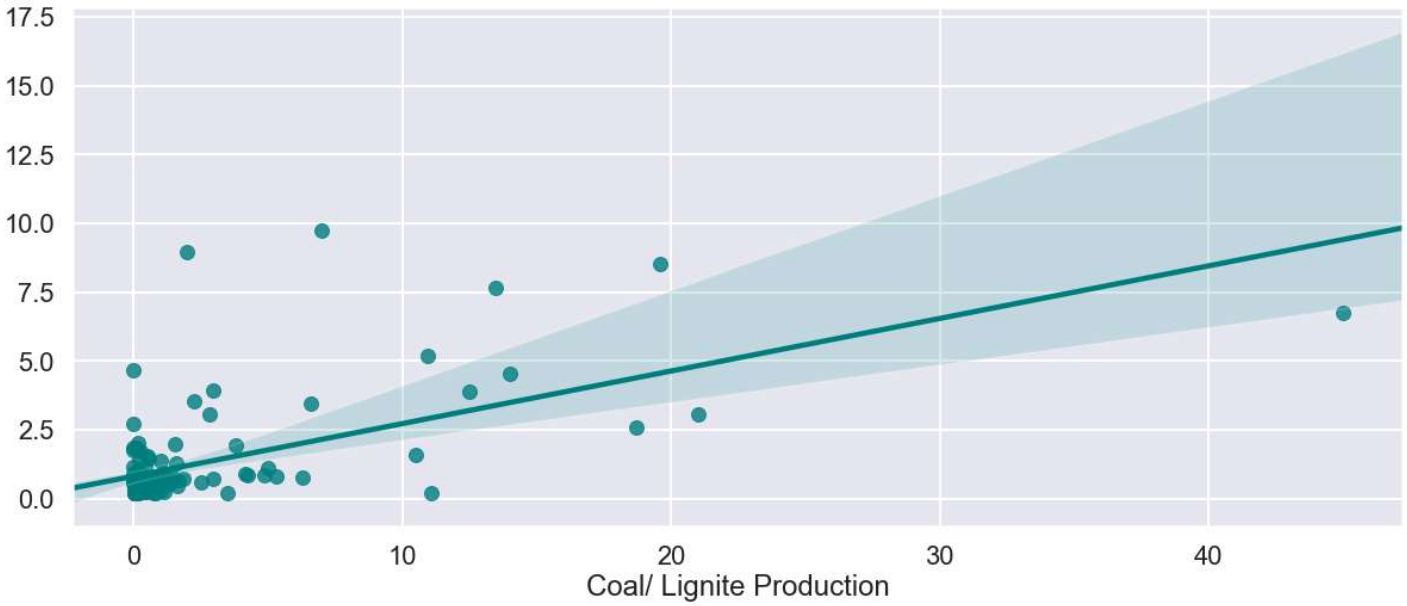
Plotting the Best fit Line

In [210…

```
plt.figure(figsize=(20,8))
sns.set_context('poster')
sns.regplot(y_test,predict, color ='teal')
```

Out[210…

```
<matplotlib.axes._subplots.AxesSubplot at 0x17281453470>
```



Let's Now Look for the Errors

In [ ]:

In [204…
```
from sklearn import metrics
```

In [206…
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, predict))
```

```
Mean Absolute Error: 1.7866185961352241
```

In [209…
```
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, predict)))
```

```
Root Mean Squared Error: 4.557226397041358
```

In [ ]:
```
# And we're Done !
```