COMBINED-CYCLE POWER PLANT

DATA SCIENCE PROJECT

PROJECT DETAILS

TEAM MEMBERS

VAISHNAVI N	MANPREET BANJA	VAIBHAV HINDIA	ANKIT YADAV	MEGHA S K	SUSHRUTHA UK

ights reserved.

Thesis Presentation Outline

Free SlideSalad PowerPoint Template

slidesalad

Copyright (c) slidesalad.com All rights reserved

INTRODUCTION

- A combined-cycle power plant comprises of gas turbines, steam turbines, and heat recovery steam generators.
- In this type of plant, the electricity is generated by gas and steam turbines combined in one cycle. Then, it is transferred from one turbine to another.
- A Combined cycle power plant is a highly efficient power generation unit. They are clean and highly efficient.
- The process of combined cycle power generation recovers the temperature from the exhaust gas and utilizes that heat in power generation.

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

BUSINESS MODEL OBJECTIVE

The objective of the project is that we have to model the energy generated as a function of exhaust vacuum and ambient variables and use that model to improve the plant's performance.

This is a project where the variable to be predicted is energy production.

Free SlideSalad PowerPoint Template

Copyright (c) SlideSalad.com All rights reserved

PROJECT ARCHITECTURE

01 Collection of data

02 Business understanding

03 Exploratory Data Analysis (EDA)

04 Model Building

05 Deployment

Free SlideSalad PowerPoint Template

slidesalad

Copyright (c) slidesalad.com All rights reserved

Data Set Details

Areas that you may wish to cover

The data file contains 9568 observations with five variables collected from a combined cycle power plant over six years when the power plant was set to work with a full load. The variables are the following:

- Temperature (in degrees Celsius).
- Ambient pressure (in millibar).
- Exhaust vacuum (in cm Hg).
- Relative humidity (in percentage).
- Energy Production (in MW) net hourly electrical energy output.

Copyright (c) SlideSalad.com All rights reserved

Exploratory Data Analysis (EDA)

In [2]:
```python
#Loading the Dataset
df = pd.read_csv('energy_production.csv')
df.head()
```

Out[2]:

	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
0	9.59	38.56	1017.01	60.10	481.30
1	12.04	42.34	1019.72	94.67	465.36
2	13.87	45.08	1024.42	81.69	465.48
3	13.72	54.30	1017.89	79.08	467.05
4	15.14	49.64	1023.78	75.00	463.58

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

```
In [3]: #Checking for Null values
        df.isnull().sum()

Out[3]: temperature          0
        exhaust_vacuum       0
        amb_pressure         0
        r_humidity           0
        energy_production    0
        dtype: int64
```

INFERENCE :There are no NULL Values in the dataset

```
In [4]: #Shape of Dataset
        df.shape

Out[4]: (9568, 5)

In [5]: #Data types of variables
        df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 9568 entries, 0 to 9567
        Data columns (total 5 columns):
         #   Column             Non-Null Count  Dtype  
        ---  ------             --------------  -----  
         0   temperature        9568 non-null   float64
         1   exhaust_vacuum     9568 non-null   float64
         2   amb_pressure       9568 non-null   float64
         3   r_humidity         9568 non-null   float64
         4   energy_production  9568 non-null   float64
        dtypes: float64(5)
        memory usage: 373.9 KB
```

INFERENCE: There are no NULL Values in the dataset.

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

CHECKING FOR DUPLICATE VALUES

```
df=Ep.drop_duplicates()
df
```

	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
0	9.59	38.56	1017.01	60.10	481.30
1	12.04	42.34	1019.72	94.67	465.36
2	13.87	45.08	1024.42	81.69	465.48
3	13.72	54.30	1017.89	79.08	467.05
4	15.14	49.64	1023.78	75.00	463.58
...	...	...	...	...	...
9563	17.10	49.69	1005.53	81.82	457.32
9564	24.73	65.34	1015.42	52.80	446.92
9565	30.44	56.24	1005.19	56.24	429.34
9566	23.00	66.05	1020.61	80.29	421.57
9567	17.75	49.25	1020.86	63.67	454.41

9527 rows × 5 columns

```
df.shape
```

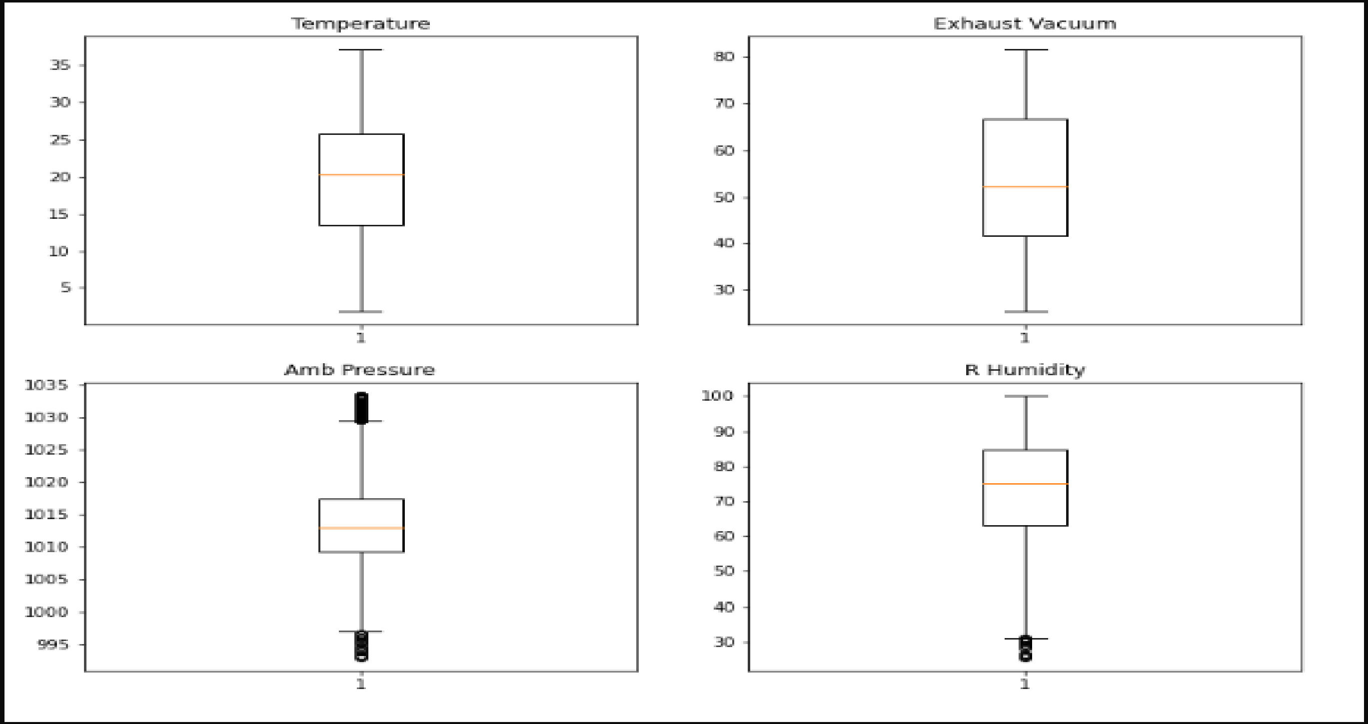
(9527, 5)

Free SlideSalad PowerPoint Template

slidesalad

Copyright (c) Slidesalad.com All rights reserved

VISUALIZATIONS
CHECKING FOR OUTLIERS USING BOXPLOT



Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

REMOVING OUTLIERS USING IQR METHOD

```
# Calculating IQR
IQR_amb = 1017.20 - 1009.08
IQR_r = 84.85 - 63.37
#printing
print("IQR for amb_pressure is: ",IQR_amb)
print("IQR for r_humidity is: ",IQR_r)

#Calculating Lower & Upper Extreme for amb_pressure
LE_amb = 1009.08 - IQR_amb * 1.5
UE_amb = 1017.20 + IQR_amb * 1.5
#printing
print("Lower Extreme of amb_pressure is: ",LE_amb)
print("Upper Extreme of amb_pressure is: ",UE_amb)

#Calculating Lower & Upper Extreme for r_humidity
LE_r = 63.37 - IQR_r * 1.5
UE_r = 84.85 + IQR_r * 1.5
#printing
print("Lower Extreme of r_humidity is: ",LE_r)
print("Upper Extreme of r_humidity is: ",UE_r)
```

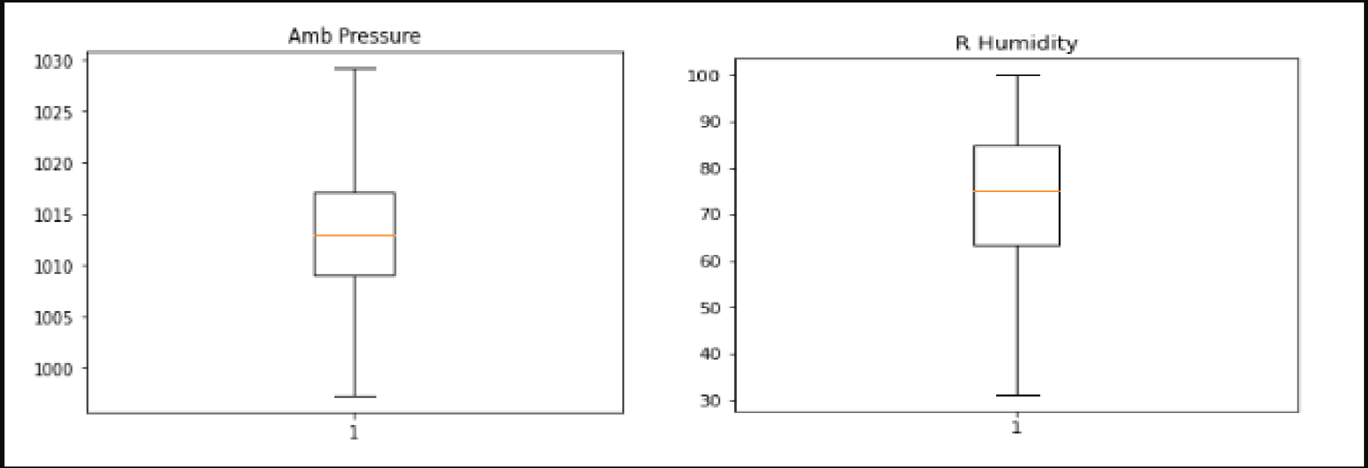
```
IQR for amb_pressure is:  8.120000000000005
IQR for r_humidity is:  21.479999999999997
Lower Extreme of amb_pressure is:  996.9000000000001
Upper Extreme of amb_pressure is:  1029.38
Lower Extreme of r_humidity is:  31.15
Upper Extreme of r_humidity is:  117.07
```

Free SlideSalad PowerPoint Template

slidesalad

Copyright (c) Slidesalad.com All rights reserved

AFTER REMOVING OUTLIERS



Inference : There are No more Outliers present in the dataset

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

FINAL DATA SET AFTER EDA

In [24]:
```
# Final Dataset after EDA
df5.head()
```

Out[24]:

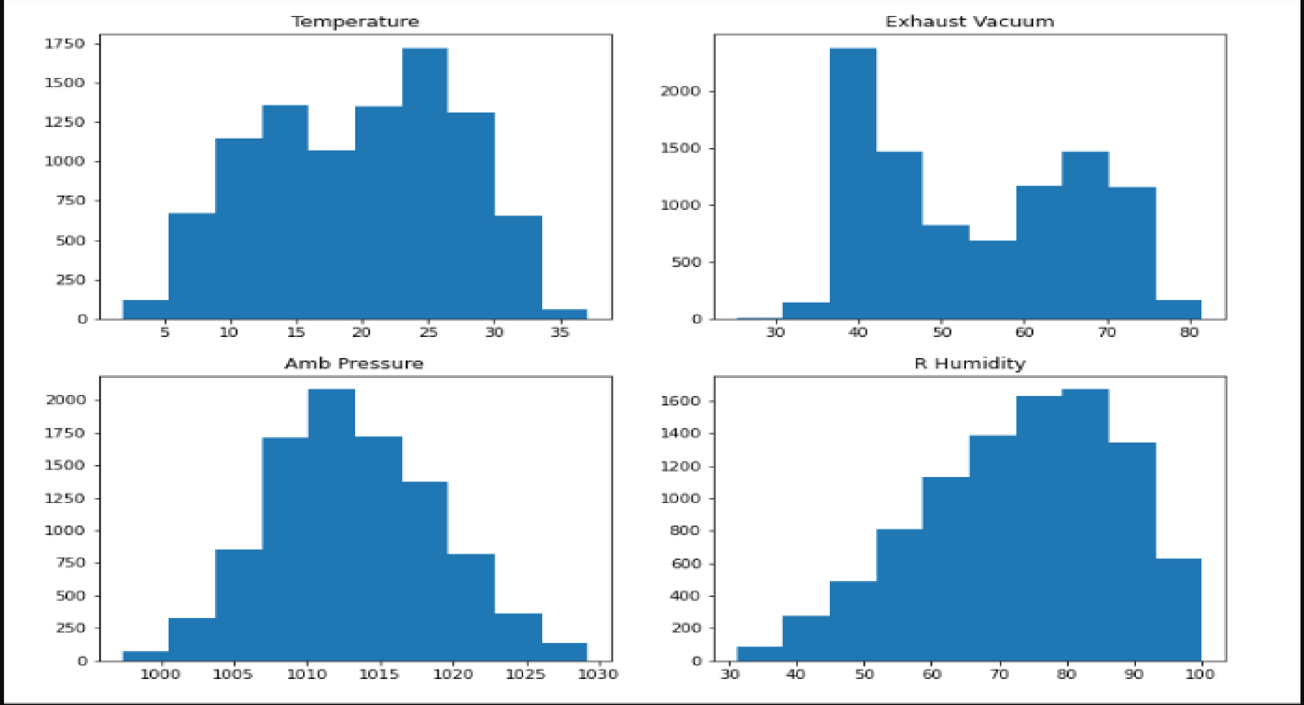
	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
0	9.59	38.56	1017.01	60.10	481.30
1	12.04	42.34	1019.72	94.67	465.36
2	13.87	45.08	1024.42	81.69	465.48
3	13.72	54.30	1017.89	79.08	467.05
4	15.14	49.64	1023.78	75.00	463.58

Free SlideSalad PowerPoint Template

slidesalad

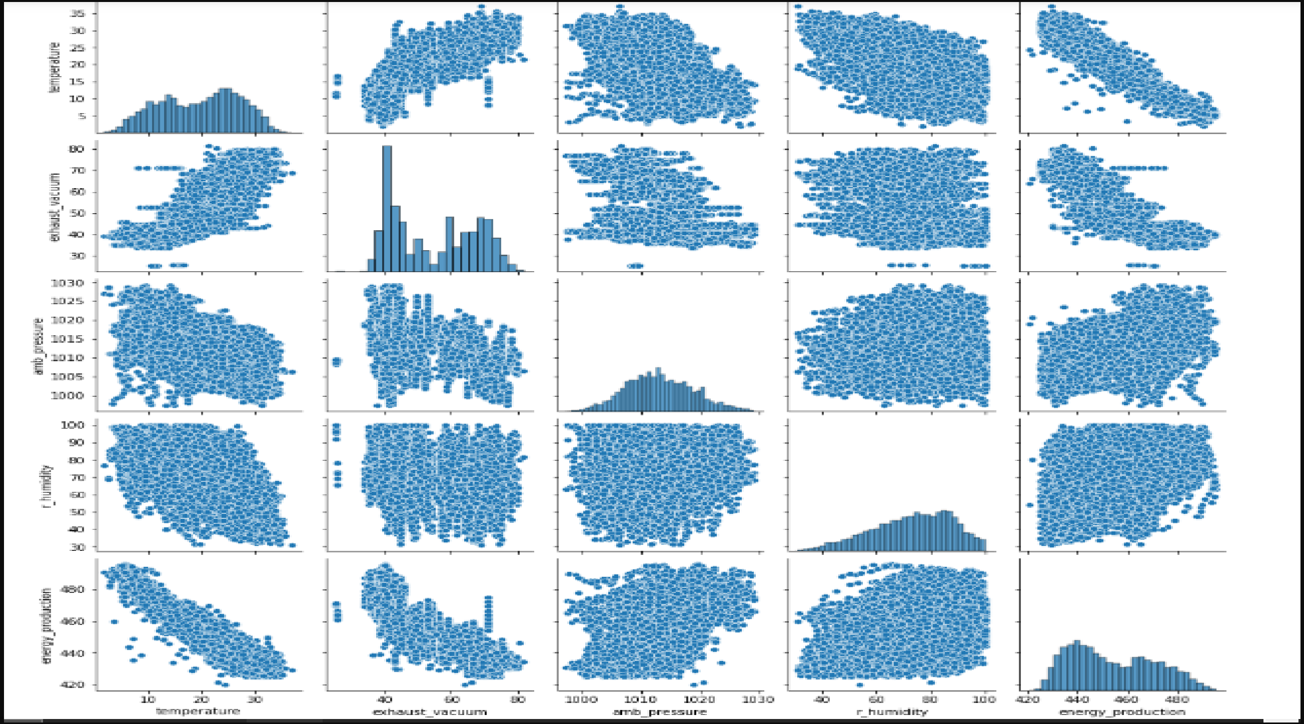
Copyright (c) SlideSalad.com All rights reserved

HISTOGRAM PLOT



Copyright (c) SlideSalad.com All rights reserved

CHECKING COLLINEARITY B/W INDEPENDENT VARIABLES



Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) slidesalad.com All rights reserved

Inference:
As we can see "temperature" & "exhaust_vaccum" have a strong positve Correlation, So we can say that there is a multicollinearity effect present.

In [27]:
```
df5.corr()
```

Out[27]:

	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
temperature	1.000000	0.842728	-0.508625	-0.542175	-0.947491
exhaust_vacuum	0.842728	1.000000	-0.415389	-0.310217	-0.868693
amb_pressure	-0.508625	-0.415389	1.000000	0.105210	0.521194
r_humidity	-0.542175	-0.310217	0.105210	1.000000	0.388023
energy_production	-0.947491	-0.868693	0.521194	0.388023	1.000000

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

CROSS CHECK WITH VIF

In [28]:
```python
rsq_Tem = smf.ols('temperature~exhaust_vacuum+amb_pressure+r_humidity',data=df5).fit().rsquared
vif_Tem = 1/(1-rsq_Tem)

rsq_ex = smf.ols('exhaust_vacuum~temperature+amb_pressure+r_humidity',data=df5).fit().rsquared
vif_ex = 1/(1-rsq_ex)

rsq_amb = smf.ols('amb_pressure~temperature+exhaust_vacuum+r_humidity',data=df5).fit().rsquared
vif_amb = 1/(1-rsq_amb)

rsq_rh = smf.ols('r_humidity~temperature+exhaust_vacuum+amb_pressure',data=df5).fit().rsquared
vif_rh = 1/(1-rsq_rh)

# Storing vif values in a data frame
d1 = {'Variables':['temperature','exhaust_vacuum','amb_pressure','r_humidity'],'VIF':[vif_Tem,vif_ex,vif_amb,vif_rh]}
Vif_frame = pd.DataFrame(d1)
Vif_frame
```

Out[28]:

	Variables	VIF
0	temperature	5.906452
1	exhaust_vacuum	3.907032
2	amb_pressure	1.447254
3	r_humidity	1.696703

Inference - There is no multi-collinearity between the variables , since VIF is less than 20.

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

OVERVIEW REPORT OF EDA

In [52]:
```
EDA_report= pp.ProfileReport(df5)
EDA_report.to_file(output_file='report.html')
```

```
Summarize dataset:   0%|           | 0/5 [00:00<?, ?it/s]
Generate report structure:   0%|           | 0/1 [00:00<?, ?it/s]
Render HTML:   0%|           | 0/1 [00:00<?, ?it/s]
Export report to file:   0%|           | 0/1 [00:00<?, ?it/s]
```

In [ ]:

In [31]:
```
df5
```

Out[31]:

	temperature	exhaust_vacuum	amb_pressure	r_humidity	energy_production
0	9.59	38.56	1017.01	60.10	481.30
1	12.04	42.34	1019.72	94.67	465.36
2	13.87	45.08	1024.42	81.69	465.48
3	13.72	54.30	1017.89	79.08	467.05
4	15.14	49.64	1023.78	75.00	463.58
...	...	...	...	...	...
9563	17.10	49.69	1005.53	81.82	457.32
9564	24.73	65.34	1015.42	52.80	446.92
9565	30.44	56.24	1005.19	56.24	429.34
9566	23.00	66.05	1020.61	80.29	421.57
9567	17.75	49.25	1020.88	63.67	454.41

9461 rows × 5 columns

Free SlideSalad PowerPoint Template

slidesalad

Copyright (c) SlideSalad.com All rights reserved

INFERENCE

- Original data set contains - 9568 observations

- Duplicates – 41 observations

- After Removal of Duplicates - 9527 observations

- After Removal of Outliers - 9417

Final Data set Contains – 9418 Observations

Free SlideSalad PowerPoint Template

Copyright (c) SlideSalad.com All rights reserved

MODEL BUILDING

MODELS USED FOR MODEL BUILDING -

- LINEAR REGRESSION
- LASSO AND RIDGE REGRESSION
- DECISION TREE REGRESSION
- RANDOM FOREST REGRESSION
- ADABOOST REGRESSOR
- KNN MODEL
- XGBoost Regressor

Free SlideSalad PowerPoint Template

Copyright (c) SlideSalad.com All rights reserved

SPLITTING X and Y

01.
LINEAR
MODEL

```
#Splitting X & Y Variable
# X Valable
X = df5.drop(['energy_production'], axis = 'columns')
# Y Variable
Y = df5.energy_production
#printing shape of both
print("X :",X.shape)
print("Y :",len(Y))
```

```
X : (9417, 4)
Y : 9417
```

```
X_train.shape
```

```
(7533, 4)
```

```
X_test.shape
```

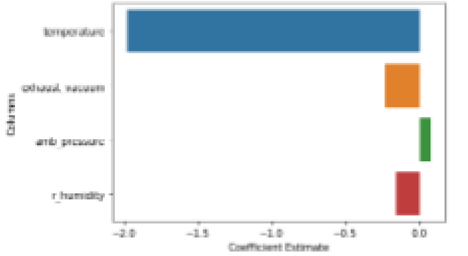
```
(1884, 4)
```

Linear model

```
#Model
Lmodel = LinearRegression()
Lmodel.fit(X_train,Y_train)
Lmodel.score(X_test,Y_test)*100
```

```
92.50970525865723
```

<AxesSubplot:xlabel='Coefficient Estimate', ylabel='Columns'>



Copyright (c) Slidesalad.com All rights reserved.

02.
LASSO & REGRESSION

```python
# Ridge regression
from sklearn.linear_model import Ridge

# Train the model
ridgeR = Ridge(alpha = 1)
ridgeR.fit(X_train, Y_train)
y_pred = ridgeR.predict(X_test)

# calculate mean square error
mean_squared_error_ridge = np.mean((y_pred - Y_test)**2)
print("Mean_Squared_error:",mean_squared_error_ridge)

# get ridge coefficient and print them
ridge_coefficient = pd.DataFrame()
ridge_coefficient["Columns"]= X_train.columns
ridge_coefficient['Coefficient Estimate'] = pd.Series(ridgeR.coef_)
print(ridge_coefficient)
```

```
Mean_Squared_error: 21.01452500349596
         Columns  Coefficient Estimate
0    temperature             -1.976951
1  exhaust_vacuum            -0.230662
2    amb_pressure             0.080297
3      r_humidity            -0.160455
```

```python
# Lasso Regression
from sklearn.linear_model import Lasso

# Train the model
lasso = Lasso(alpha = 1)
lasso.fit(X_train, Y_train)
y_pred1 = lasso.predict(X_test)

# Calculate Mean Squared Error
mean_squared_error = np.mean((y_pred1 - Y_test)**2)
print("Mean squared error on test set", mean_squared_error)
lasso_coeff = pd.DataFrame()
lasso_coeff["Columns"] = X_train.columns
lasso_coeff['Coefficient Estimate'] = pd.Series(lasso.coef_)

print(lasso_coeff)
```

```
Mean squared error on test set 21.021888475894897
         Columns  Coefficient Estimate
0    temperature             -1.921902
1  exhaust_vacuum            -0.248816
2    amb_pressure             0.064529
3      r_humidity            -0.144717
```

Free SlideSalad PowerPoint Template

Copyright (c) Slidesalad.com All rights reserved

03. DECISION TREE REGRESSOR

```
# Decision Tree Regressor
seed = 7
kfold = KFold(n_splits=10, random_state=seed,shuffle=True)
cart = DecisionTreeRegressor()
num_trees = 100
model = BaggingRegressor(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, X_train, Y_train, cv=kfold)
print(results.mean())
```

```
0.9599123064600011
```

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

04. RANDOM FOREST REGRESSOR

```python
# Random Forest Regressor
num_trees = 100
max_features = 3
kfold = KFold(n_splits=10, random_state=7,shuffle=True)
model = RandomForestRegressor(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, X_train, Y_train, cv=kfold)
print(results.mean())
```

```
0.9607671535736031
```

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

05. ADABOOST REGRESSOR

```python
# Adaboost Regressor
num_trees = 10
seed=7
kfold = KFold(n_splits=10, random_state=seed,shuffle=True)
model = AdaBoostRegressor(n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, X_train, Y_train, cv=kfold)
print(results.mean())
```

```
0.907573194628539
```

Free SlideSalad PowerPoint Template

Copyright (c) slidesalad.com All rights reserved

06.
XGBOOST REGRESSOR

```
F_model = XGBRegressor(n_estimators=250,learning_rate=0.2, max_depth=5)
F_model.fit(X_train,Y_train)
F_model.score(X_test,Y_test)*100
```

```
96.30225774714421
```

Free SlideSalad PowerPoint Template

slidesalad

Copyright (c) SlideSalad.com All rights reserved

07. KNN MODEL

```
n_neighbors = np.array(range(1,41))
param_grid = dict(n_neighbors=n_neighbors)
model = KNeighborsRegressor()
grid = GridSearchCV(estimator=model,param_grid=param_grid)
grid.fit(X_train,Y_train)
print(grid.best_score_)
print(grid.best_params_)
```

```
0.9436345358354734
{'n_neighbors': 5}
```

```
KNN_model = KNeighborsRegressor(n_neighbors=5)
KNN_model.fit(X_train,Y_train)
```

```
KNeighborsRegressor()
```

```
Y_pred = KNN_model.predict(X_test)
KNN_1 = metrics.explained_variance_score(Y_test,Y_pred)
print("Accuracy:" ,KNN_1)
```

```
Accuracy: 0.9443999431784652
```

Free SlideSalad PowerPoint Template
slidesalad
Copyright (c) SlideSalad.com All rights reserved

CHOOSING THE BEST MODEL FOR DEPLOYMENT

```
In [45]: def find_best_model_using_gridsearchcv(x,y):
             algos = {
                 'linear_regression' :{'model': LinearRegression(),'params':{'normalize': [False]}
                 },
                 'ridge': {'model': Ridge(),'params': { 'alpha': [0.1,0.1,0.5,1]}
                 },
                 'lasso': {'model': Lasso(),'params': {'alpha': [0.1,0.5,1],'selection': ['random', 'cyclic']}
                 },
                 'decision_tree': {'model': DecisionTreeRegressor(),'params': {'criterion' : ['mse', 'friedman_mse'],'splitter': ['best',
                 },
                 'Random Forest': {'model': RandomForestRegressor(),'params': {'n_estimators' : [100,125,150,200],'max_features': [3,4]}
                 },
                 'XGBoost': {'model': XGBRegressor(),'params': {'n_estimators' : [100,125,150,200,225,250],'max_depth': [3,4,5],'learning_
                 },
                 'SVM': {'model': SVR(),'params': {'kernel' :['rbf']}
                 },
                 'KNN': {'model': KNeighborsRegressor(),'params': {'n_neighbors':[5,10,15,20,25,30,35,40,45,50]}
                 }
             }
             scores = []
             cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=10)
             for algo_name, config in algos.items():
                 gs =  GridSearchCV(config['model'], config['params'], cv=cv)
                 gs.fit(x,y)
                 scores.append({
                     'model': algo_name,
                     'best_score': gs.best_score_,
                     'best_params': gs.best_params_
                 })

             return pd.DataFrame(scores,columns=['model','best_score','best_params'])

         find_best_model_using_gridsearchcv(X,Y)
```


Copyright (c) slidesalad.com All rights reserved

THE FINAL MODEL CHOSEN -

RANDOM FOREST MODEL

	model	best_score	best_params
0	linear_regression	0.929326	{'normalize': False}
1	ridge	0.929326	{'alpha': 1}
2	lasso	0.929328	{'alpha': 0.1, 'selection': 'random'}
3	decision_tree	0.924775	{'criterion': 'friedman_mse', 'splitter': 'best'}
4	Random Forest	0.962977	{'max_features': 3, 'n_estimators': 200}
5	XGBoost	0.966608	{'learning_rate': 0.2, 'max_depth': 5, 'n_esti...
6	SVM	0.376926	{'kernel': 'rbf'}
7	KNN	0.945204	{'n_neighbors': 5}

ACCURACY – 96.29%

Free SlideSalad PowerPoint Template

Copyright (C) SlideSalad.com All rights reserved.

THANK YOU !