Regression Analysis

Naval Vessel Dataset

GROUP – 12

1. Manogna Bedadakota
2. Sheik Ishaaq
3. Sri Ram
4. Sandhya

B. Sc. Hons Data Science
Bhavan's Vivekananda College

PROJECT STAGES

1. DATA PRE-PROCESSING

2. EXPLORATORY DATA ANALYSIS

3. MODEL TRAINING & ACCURACY

4. COMPARING RESULTS

5. CONCLUSION & INSIGNTS

ABOUT THE DATA

The dataset has information about Naval Vessel. It contains numerical data regarding the various pressures, temperatures, turbines, etc.

OBJECTIVE

The task at hand is to apply various regression models to the data and determine the most efficient '**GT Compressor decay state coefficient**' & '**GT Turbine decay state coefficient**'.

THE PATH

Finding the best fit ML model by applying various models to the dataset.

DATA QUALITY CHECKING

No. of instances	11934
No. of attributes	18
attribute breakdown	16 input & 2 output variables
Missing values	-

ABOUT THE DATA

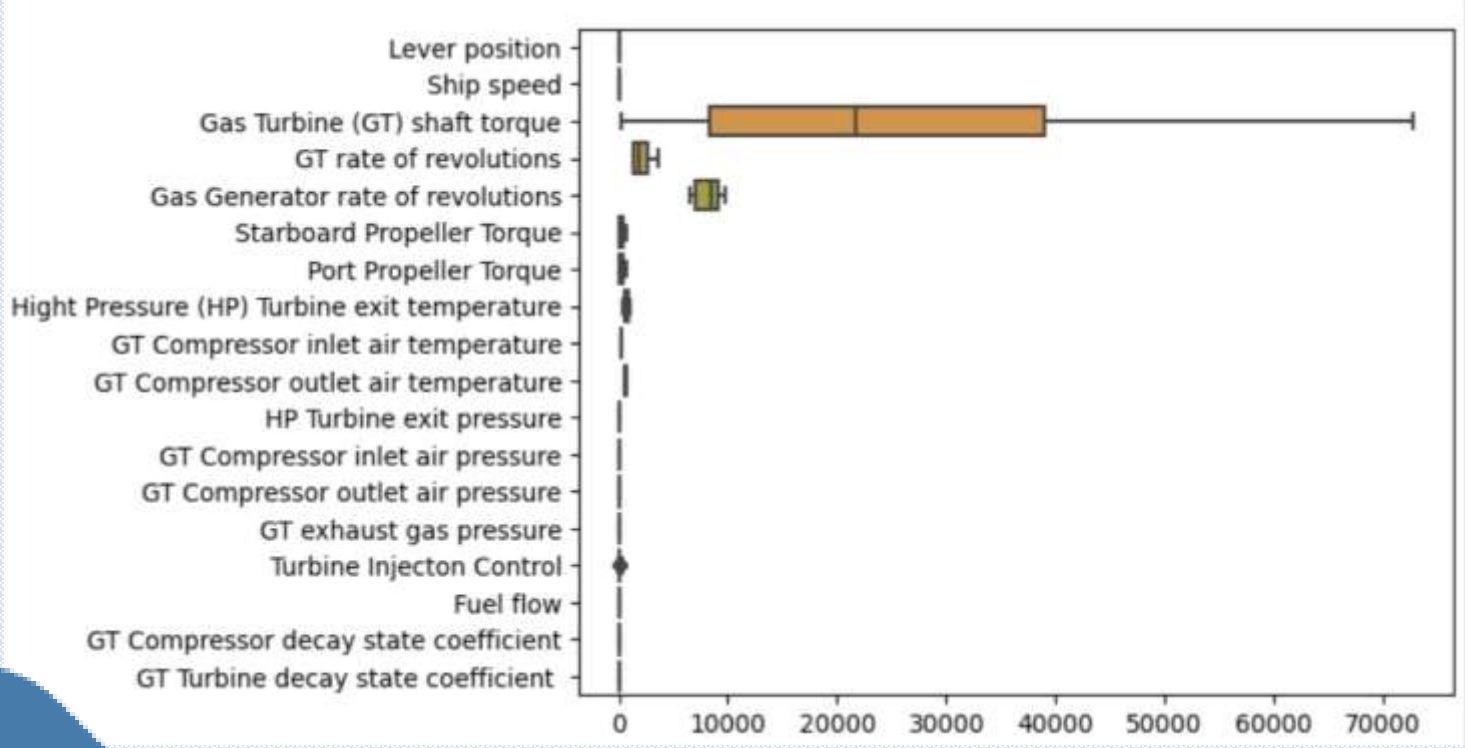
ATTRIBUTE	DESCRIPTION
Lever position	position or setting of the control lever
Ship speed	speed at which the naval vessel is traveling
GT Shaft Torque	twisting force applied to the shaft
GT rate of revolutions	rotational speed of the gas turbine engine's shaft
Generator rate of revolutions	rate of rotation of the gas generator within the GT
Starboard Propeller Torque	twisting force applied to the propeller on the right side
Port Propeller Torque	twisting force applied to the propeller on the left side
HP Turbine exit temperature	temperature of the exhaust gases leaving the high-pressure turbine
Compressor inlet air temp.	temperature of the air entering the gas turbine compressor
Compressor outlet air temp	temperature of the air exiting the gas turbine compressor
HP Turbine exit pressure	The pressure of the exhaust gases leaving the high-pressure turbine
Compressor inlet air pressure	pressure of the incoming air entering the gas turbine compressor
Compressor outlet air pressure	pressure of the air leaving the gas turbine compressor
exhaust gas pressure	pressure of the exhaust gases exiting the gas turbine
Turbine Injecton Control	control mechanism governing the injection of fuel
Fuel flow	rate at which fuel is supplied to the gas turbine

DATA PRE PROCESSING

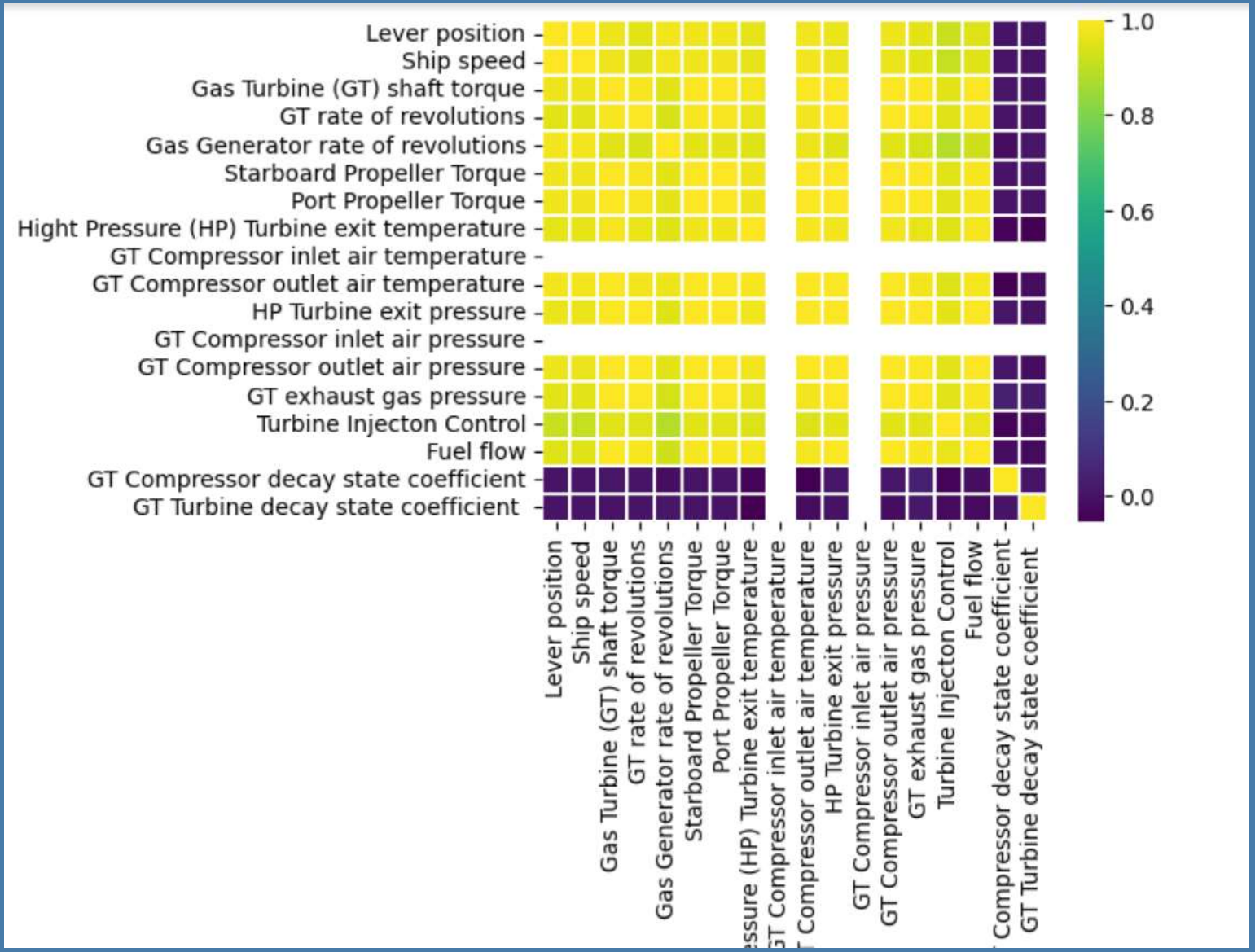
- There are 11934 rows and 18 columns.
- There are no null values
- There are 1326 outliers.
- After removing the outliers, the new data has 10608 rows and 18 columns.

EXPLORATORY DATA ANALYSIS

Box plot to check for outliers

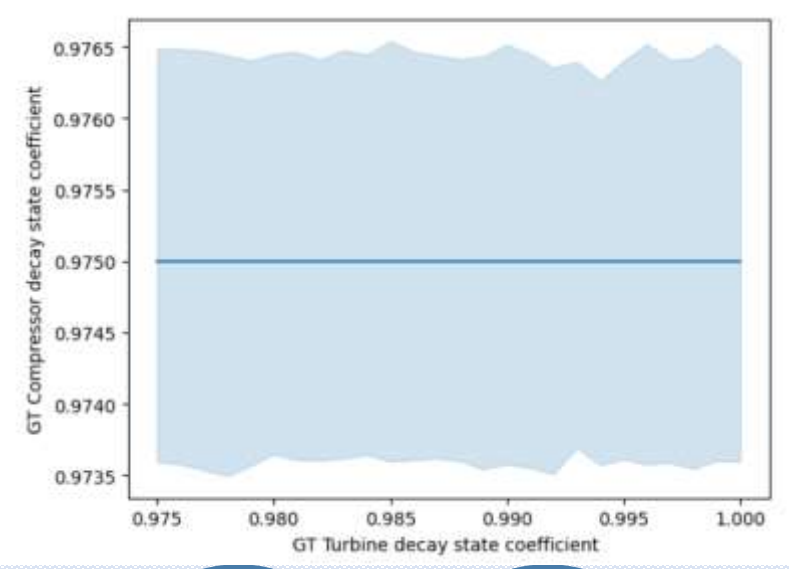


HEATMAP DISPLAYING CORELATIONS

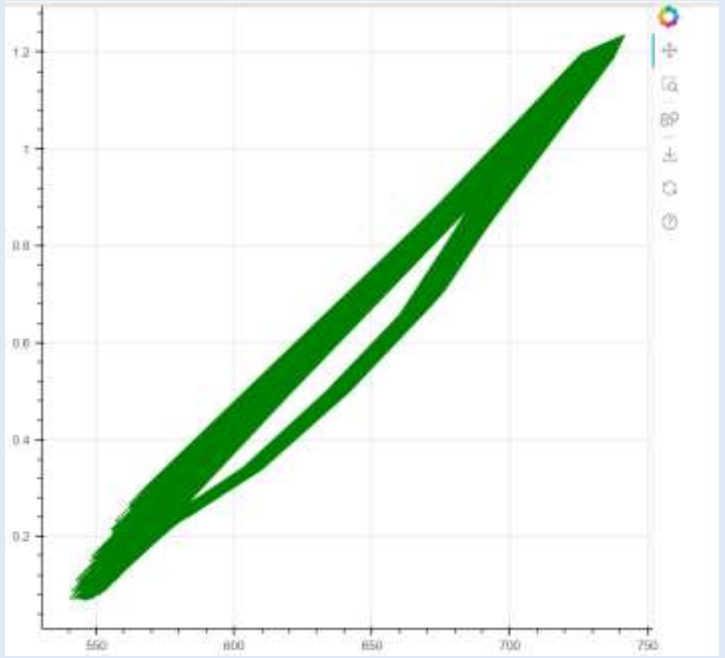


It shows positive corelation among 16/18 columns.

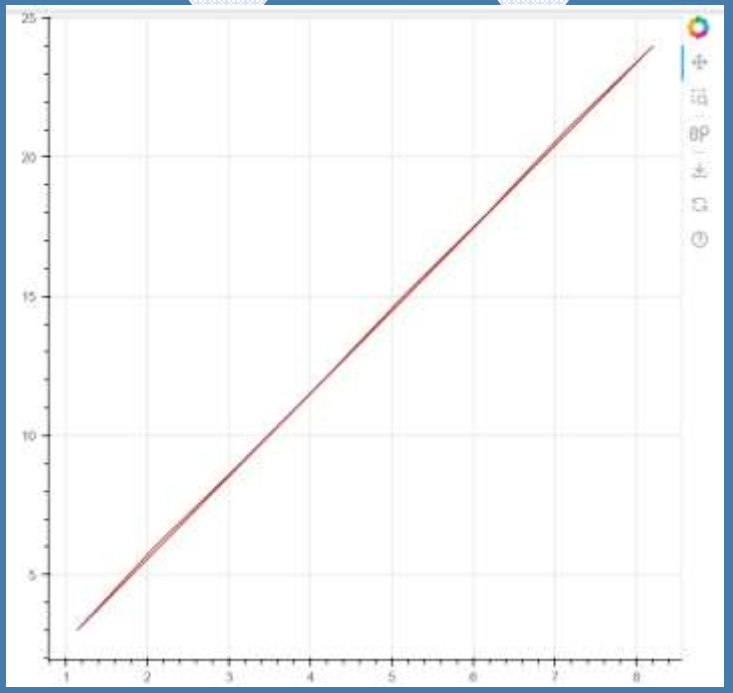
2 of the columns show no corelation.



Graph plotting the 2 target variables against each other.

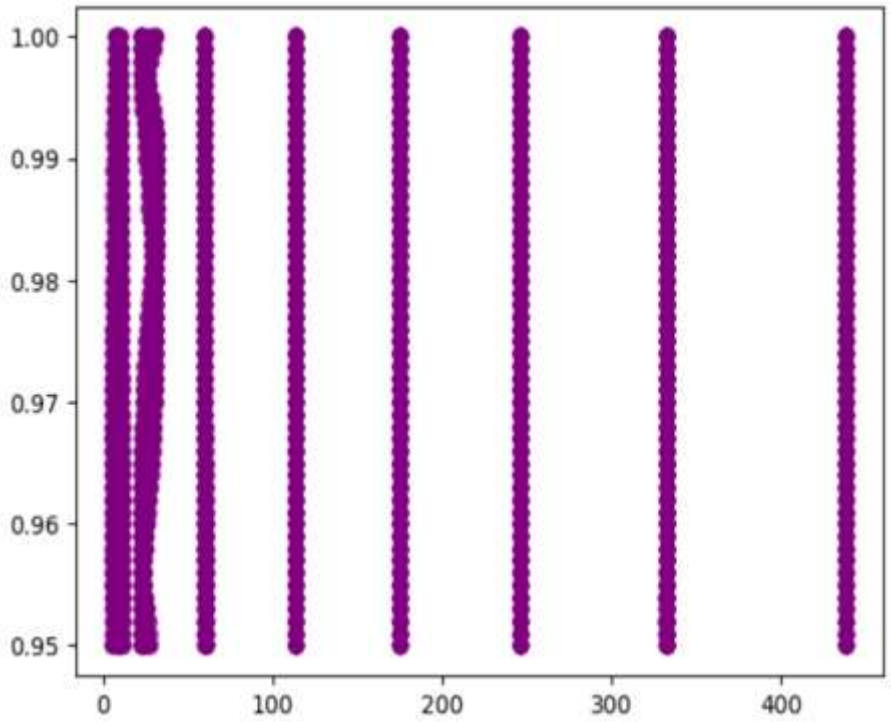


Line graph showing the relation between number of revolutions and torque.

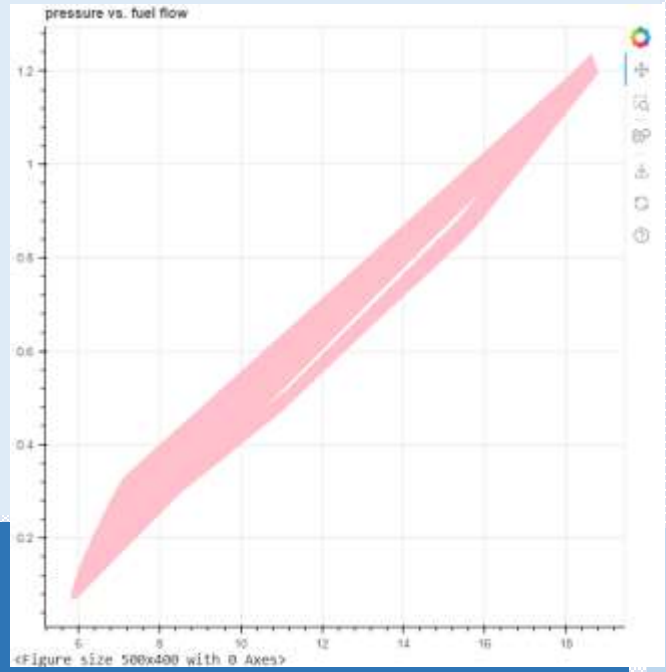


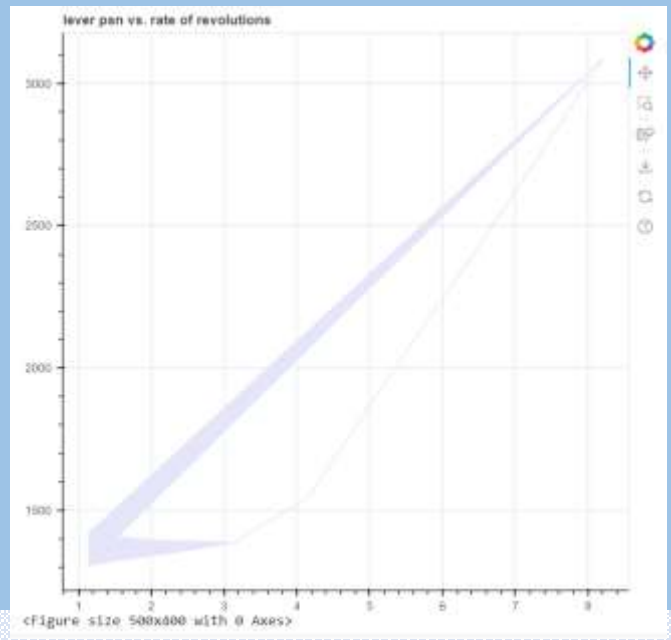
Line plot depicting relation between temperature and fuel flow

The propeller torque and GT decay coefficient plotted against each other.



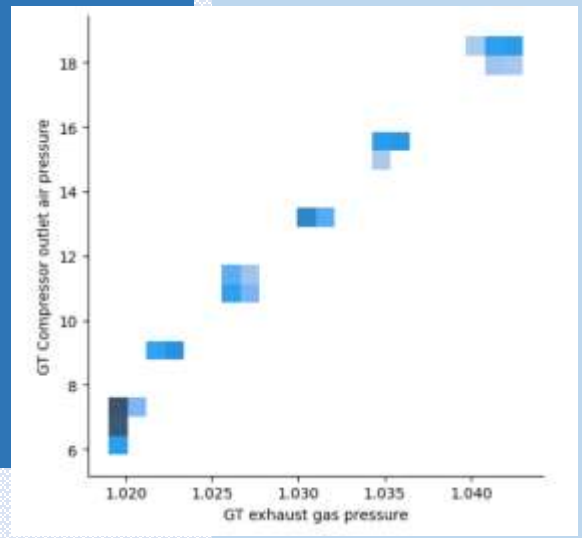
Outlet air pressure plotted against fuel flow





Relation between lever position and rate of revolutions

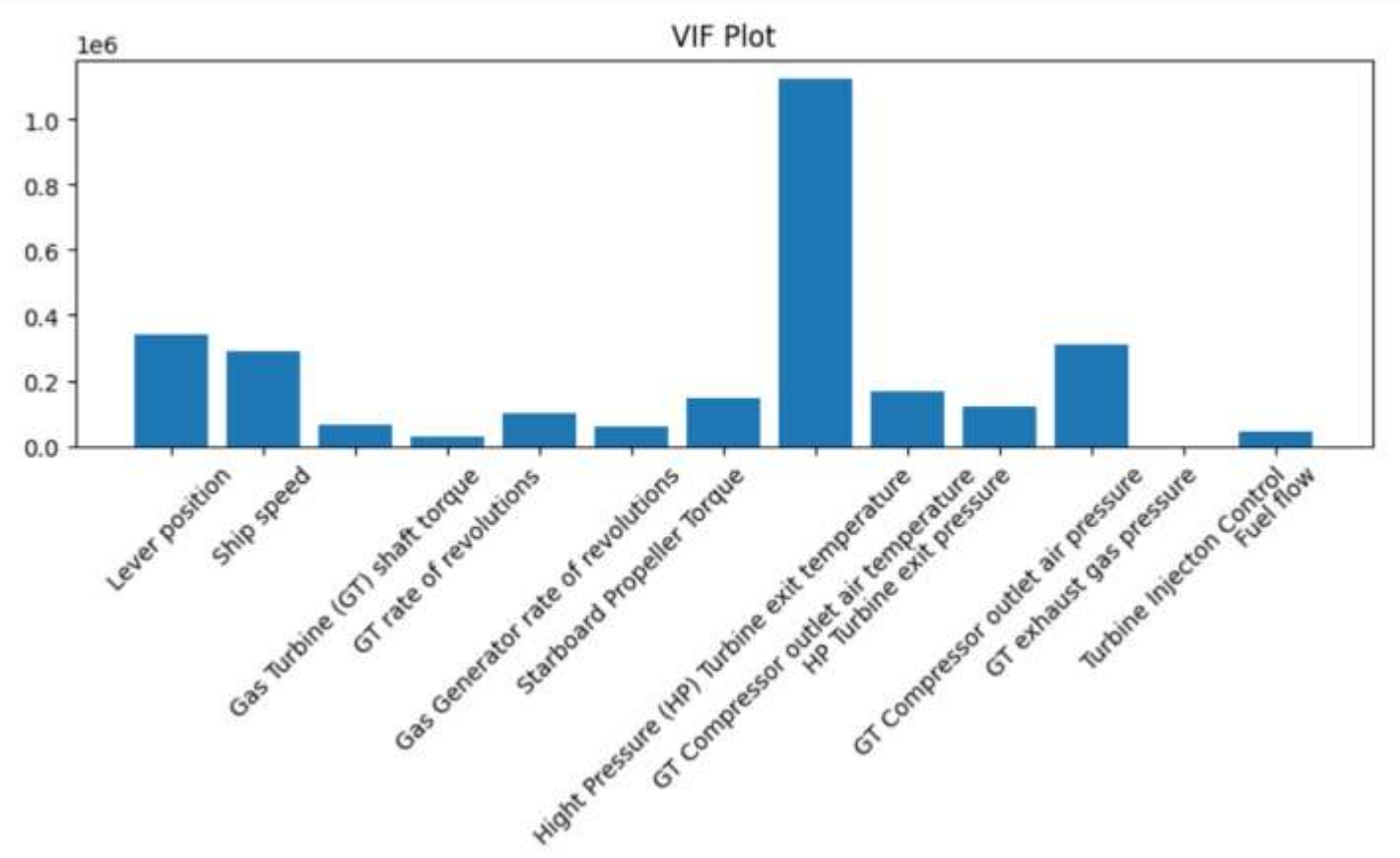
Graph of GT outlet air pressure against GT exhaust gas pressure



VARIANCE INFLATION FACTOR

	VARIABLES	VIF
0	Lever position	3.412159e+05
1	Ship speed	2.889466e+05
2	Gas Turbine (GT) shaft torque	6.450279e+04
3	GT rate of revolutions	3.001520e+04
4	Gas Generator rate of revolutions	9.843502e+04
5	Starboard Propeller Torque	5.734227e+04
6	Hight Pressure (HP) Turbine exit temperature	1.467546e+05
7	GT Compressor outlet air temperature	1.126406e+06
8	HP Turbine exit pressure	1.689188e+05
9	GT Compressor outlet air pressure	1.204855e+05
10	GT exhaust gas pressure	3.121826e+05
11	Turbine Injecton Control	2.292302e+02
12	Fuel flow	4.397066e+04

- Measure of how much the variance of an estimated regression coefficient increases when your predictors are correlated.
- Helps assess the degree of multicollinearity among independent variables.
- Tells us how much the standard error of the estimated coefficients is increased.



Bar Graph displaying the multicollinearity of columns.

It is observed that High Pressure Turbine exit temperature shows maximum collinearity and Turbine Injecton Control shows least collinearity.

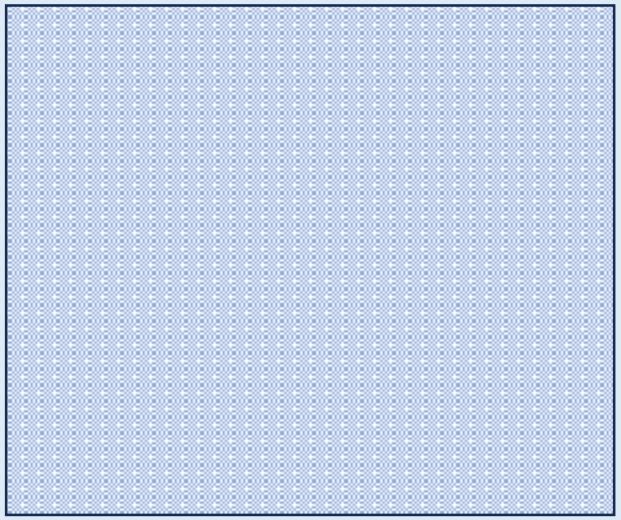
MODEL BUILDING

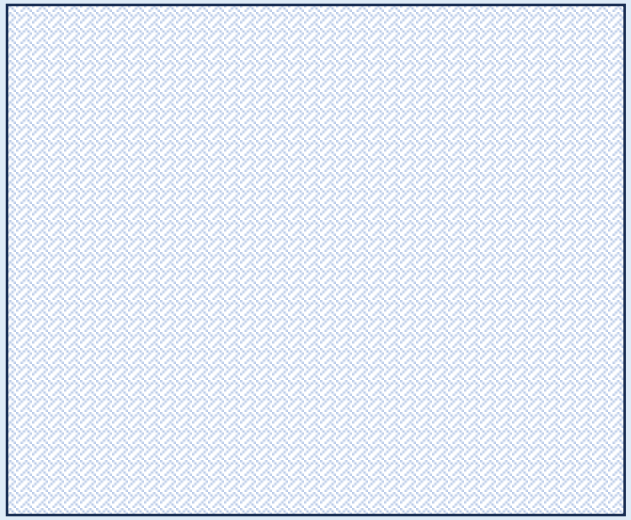
DECISION TREE REGRESSOR	
TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	4e-06
75 – 25	4e-06
80 – 20	4e-06

SUPPORT VECTOR REGRESSOR

TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	**0.000213**
75 – 25	0.000216
80 – 20	0.000219

performs regression tasks by finding a hyperplane in a high-dimensional space, which best represents the relationships between the input features and the target variable





An instance-based learning algorithm where a data point's prediction is based on the majority class of its k-nearest neighbors.

K NEIGHBOURS REGRESSOR	
TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	1.7e – 05
75 – 25	1.6e – 05
80 – 20	**1.5e - 05**

LINEAR REGRESSION	
TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	**2.8e – 05**
75 – 25	2.9e – 05
80 – 20	2.9e - 05

Linear Regression is a statistical method that models the relationship between a dependent variable and one or more independent variables by fitting a linear equation.

Gradient Boosting is an ensemble technique that builds a series of weak models (typically decision trees) sequentially, with each model trying to correct the errors of the previous one.

GRADIENT BOOST REGRESSOR	
TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	9.9e – 05
75 – 25	**0.000102**
80 – 20	0.000107

RANDOM FORREST REGRESSOR	
TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	1e – 06
75 – 25	1e – 06
80 – 20	1e - 06

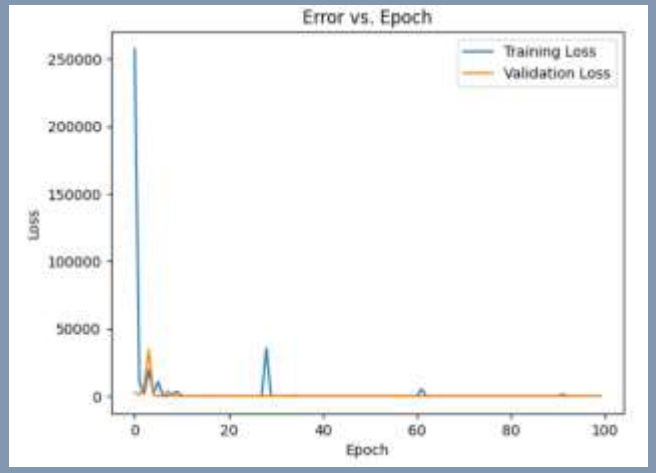
Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees.

XG BOOST REGRESSOR	
TRAIN – TEST – SPLIT	MEAN SQUARED ERROR
70 – 30	**1.3e – 05**
75 – 25	1.6e – 05
80 – 20	1.6e - 05

XGBoost is an optimized and scalable gradient boosting library that performs well in machine learning competitions; it sequentially adds weak learners to the model, focusing on correcting errors from previous iterations.

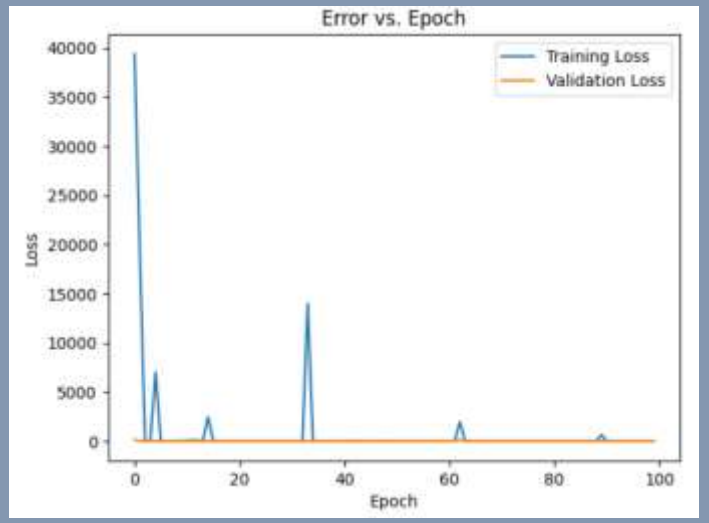
NEURAL NETWORKS

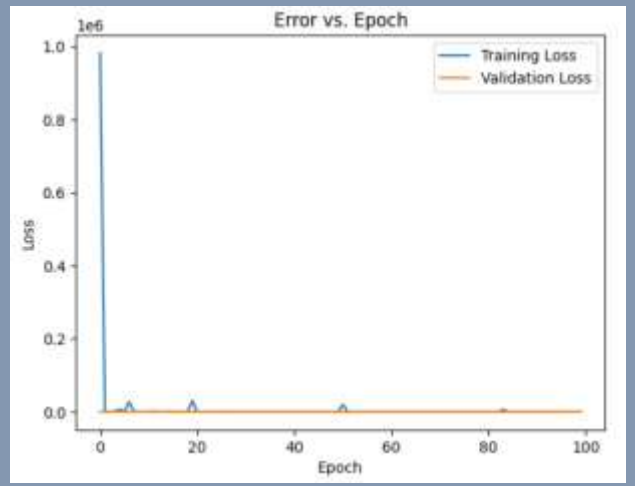
TRAIN – TEST – SPLIT	MEAN ABSOLUTE ERROR	LOSS
70 – 30	**0.030350**	**0.001689**
75 – 25	0.085742	0.008232
80 – 20	0.031832	0.001597



Training & validation loss for test split of 70 - 30

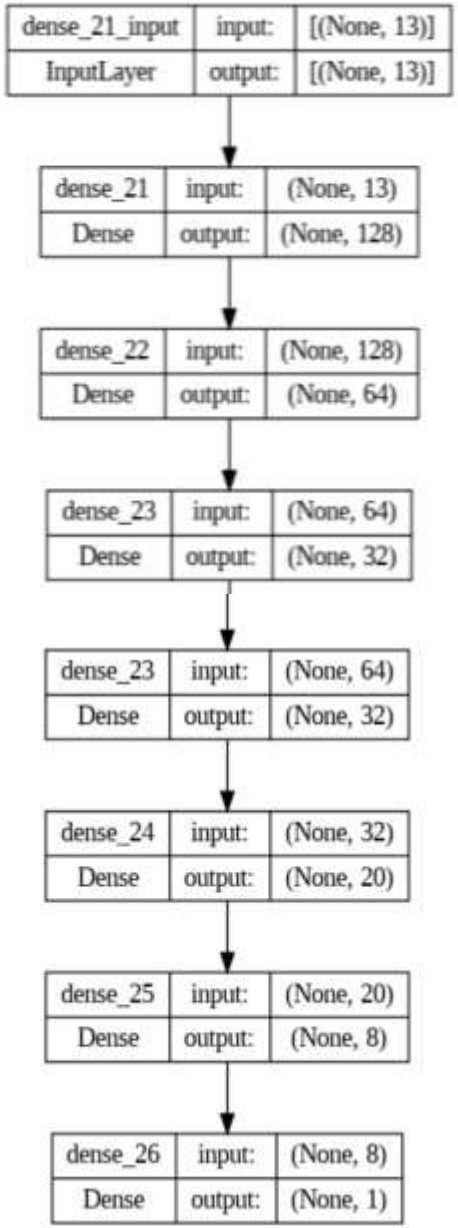
Training & validation loss for test split of 75 - 25





Training & validation loss for test split of 80 - 20

NEURAL NETWORK MODAL



COMPARISION OF MODALS

MODAL NAME	ERROR
Decision Tree	4e - 06
Support Vector Machine	0.000213
K Neighbours	1.5e - 05
Linear Regression	2.8e – 05
Gradient Boost	**0.000102**
Random Forrest	1e – 06
X G Boost	1.3e – 05
Neural Network	0.030350

CONCLUSION

- We observed that 5/8 modals have very low error. It could be an indication that the models are overfitting.
- Among the other 3 modals, Gradient Boost has the least error and could be considered as the best modal for this dataset.

APPENDIX

SPLITTING THE DATASET

```
x = new_data.drop(['GT Compressor decay state coefficient', 'GT Turbine decay state coefficient ', 'Port Propeller Torque',
                   'GT Compressor inlet air temperature', 'GT Compressor inlet air pressure'], axis=1)
y = new_data['GT Compressor decay state coefficient']

x1_train, x1_test, y1_train, y1_test = train_test_split(x, y, test_size = 0.3, random_state = 1)

x2_train, x2_test, y2_train, y2_test = train_test_split(x, y, test_size = 0.25, random_state = 1)

x3_train, x3_test, y3_train, y3_test = train_test_split(x, y, test_size = 0.2, random_state = 1)
```

```
x1_train.shape, x1_test.shape
```

```
((7425, 13), (3183, 13))
```

```
x2_train.shape, x2_test.shape
```

```
((7956, 13), (2652, 13))
```

```
x3_train.shape, x3_test.shape
```

```
((8486, 13), (2122, 13))
```

VARIANCE INFLATION FACTOR

```
def calculatevif(x):

    vif = pd.DataFrame()
    vif['VARIABLES'] = x.columns
    vif['VIF'] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
    return(vif)

calculatevif(x)
```

```
def xgregress(a, b, c, d):
    xgreg = xg.XGBRegressor(objective ='reg:linear', n_estimators = 10, seed = 123)
    xgreg.fit(a, c)
    y_pred = xgreg.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    xgmse.append(mse)
    return mse
```

XG BOOST

DECISION TREE REGRESSOR

```
def decisiontree(a, b, c, d):
    destree = DecisionTreeRegressor()
    destree.fit(a, c)
    y_pred = destree.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    desmse.append(mse)
    return mse
```

```
def supportvector(a, b, c, d):
    svreg = SVR(kernel='linear')
    svreg.fit(a, c.ravel())
    y_pred = svreg.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    svrmse.append(mse)
    return mse
```

SUPPORT VECTOR REGRESSOR

K NEIGHBORS REGRESSOR

```
def knregress(a, b, c, d):
    knreg = KNeighborsRegressor(n_neighbors=5)
    knreg.fit(a, c)
    y_pred = knreg.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    knmse.append(mse)
    return mse
```

```
def lnregress(a, b, c, d):
    lnrreg = LinearRegression()
    lnrreg.fit(a, c)
    y_pred = lnrreg.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    lnrmse.append(mse)
    return mse
```

LINEAR REGRESSOR

GRADIENT BOOST

```
def gradboost(a, b, c, d):
    gbreg = GradientBoostingRegressor(n_estimators = 20)
    gbreg.fit(a, c)
    y_pred = gbreg.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    gbmse.append(mse)
    return mse
```

```
def rfregress(a, b, c, d):
    rfreg = RandomForestRegressor(n_estimators = 20)
    rfreg.fit(a, c)
    y_pred = rfreg.predict(b)
    mse = round(mean_squared_error(d, y_pred),6)
    rfmse.append(mse)
    return mse
```

RANDOM FOREST

NEURAL NETWORK

```
def nnetwork (a, b, c , d):
    scl = StandardScaler()
    x_train = scl.fit_transform(a)
    x_test = scl.transform(b)

    model = models.Sequential([
        layers.Dense(128, activation = 'linear', input_shape=(a.shape[1],)),
        layers.Dense(64, activation='linear'),
        layers.Dense(32, activation='linear'),
        layers.Dense(20, activation='linear'),
        layers.Dense(8, activation='linear'),
        layers.Dense(1)])

    model.compile(optimizer = 'adam', loss = 'mean_squared_error', metrics = ['mae'])
    history = model.fit(a, c, epochs = 100, batch_size = 10, validation_split = 0.1)
```

```
test_mse, test_mae = model.evaluate(b, d)
nnloss.append(round((test_mse),6))
nnmae.append(round((test_mae),6))
pred = model.predict(b)
print(pred.flatten)
print(model.summary())

return test_mse, test_mae
```

```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Error vs. Epoch')
plt.show()
```

THANK YOU