Water Quality Index

Prediction and Regression Machine Learning Project

Different types of Regression Models are trained upon different forms of same dataset.

Linear Regression

K - Nearest Neighbors

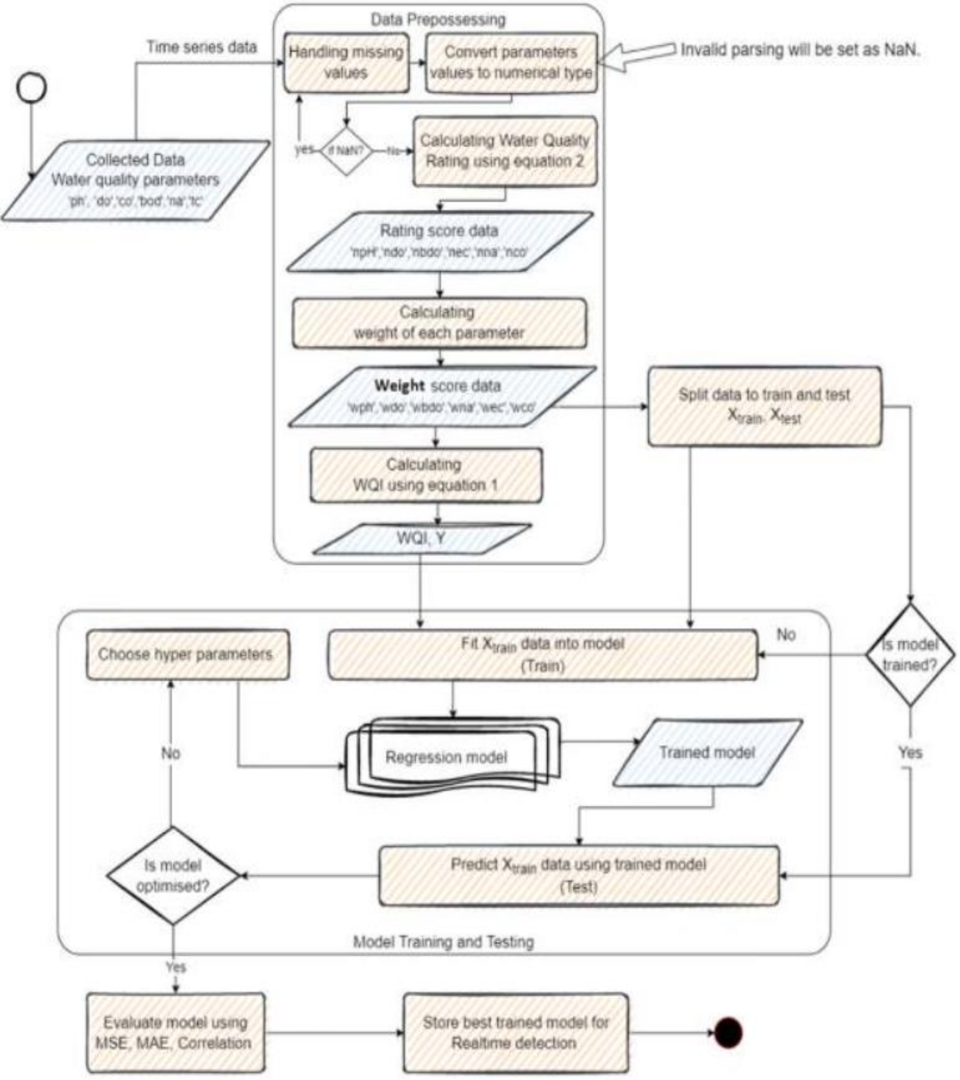
Gradient Boosting

Bayesian Ridge

Decision Tree Regression

Artificial Neural Network

Random Forest Regression



Steps involved:

1) Data Pre - Processing
2) Scaling & Normalization of values
3) Removing Outliers
4) Running the Model

```python
for col in df. columns:
    df[col].replace('?',np.NaN,inplace=True)
```

```python
df['location'].fillna(df['location'].mode()[0],inplace=True)
df['do'].fillna(df['do'].mode()[0],inplace=True)
df['ph'].fillna(df['ph'].mode()[0],inplace=True)
df['co'].fillna(df['co'].mode()[0],inplace=True)
df['bod'].fillna(df['bod'].mode()[0],inplace=True)
df['na'].fillna(df['na'].mode()[0],inplace=True)
df['tc'].fillna(df['tc'].mode()[0],inplace=True)
df['year'].fillna(df['year'].mode()[0],inplace=True)
```

All missing values are filled with NaN and then replaced with mode of the respective columns

To face the outliers present and to prevent feature domination and restricting the values between 0 and 1

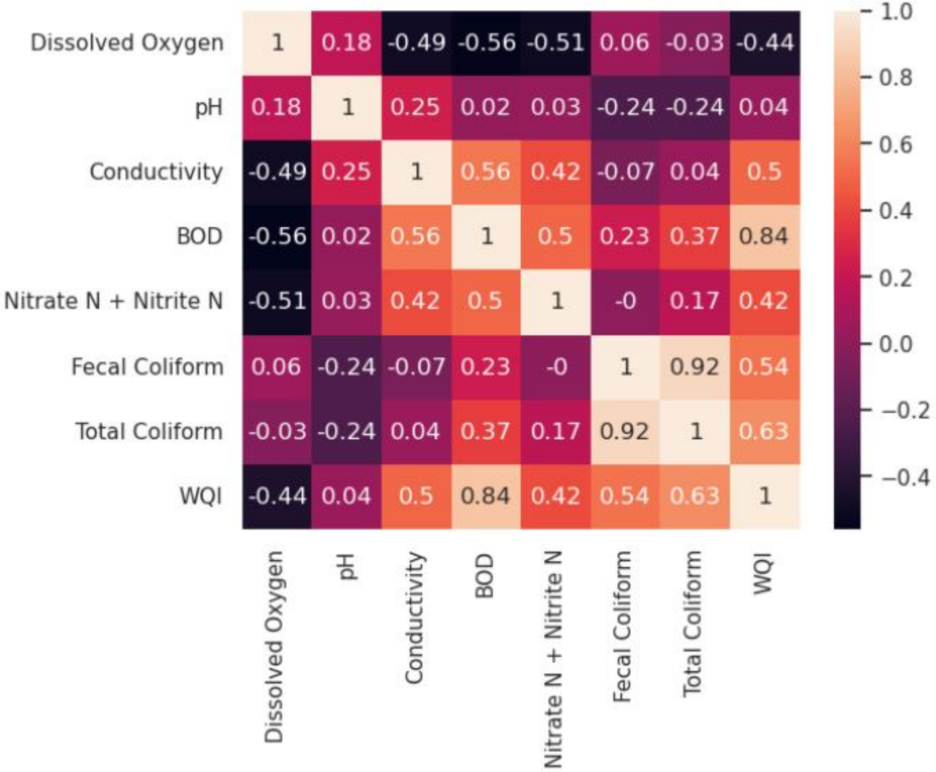
```
X_scaled = (X - X_min) / (X_max - X_min)
```

```python
#Normalizing DataSet
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Fit the scaler to the data and transform the features
X_train_scaled = scaler.fit_transform(X_train)

# Transform the testing data using the fitted scaler
X_test_scaled = scaler.transform(X_test)
```



‘Spearman’ correlation matrix

The heatmap visualization helps in understanding the strength and direction of the monotonic relationships between the variables after MinMax Scaling

```python
from scipy.stats import zscore
df_num_final_norm = zscore(df_new, axis=0)


def indices_of_greater_than_3(df_norm):
    df_norm = pd.DataFrame(df_norm)
    indices_arr = []
    n_col = df_norm.shape[1]
    for index in range(n_col):
        col_index = df_norm.iloc[: ,index]
        greater_than_3 = df_norm[col_index > 3]
        greater_than_3_index = greater_than_3.index
        indices_arr.extend(greater_than_3_index)
    return indices_arr

indices_arr = indices_of_greater_than_3(df_num_final_norm)
print("Number of outliers using Z-Score method-",len(indices_arr))
df_new.iloc[indices_arr, :]
```

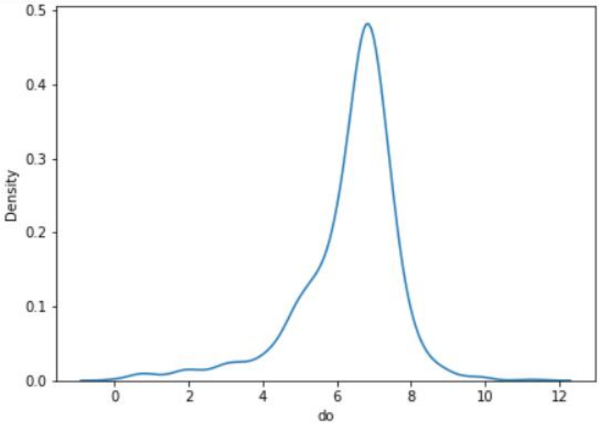
```
Number of outliers using Z-Score method- 114
```

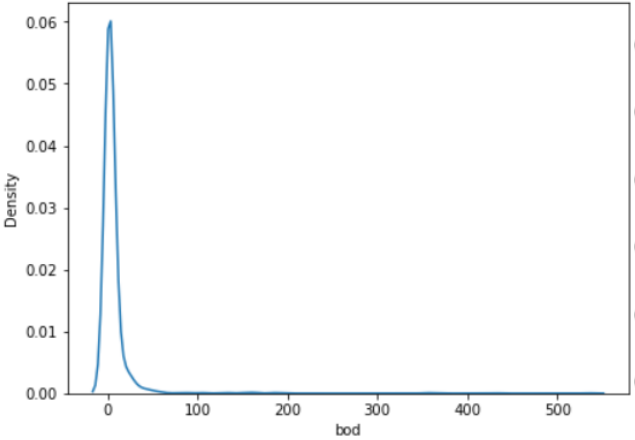
Handling outliers to remove misleading data and bias in modeling.

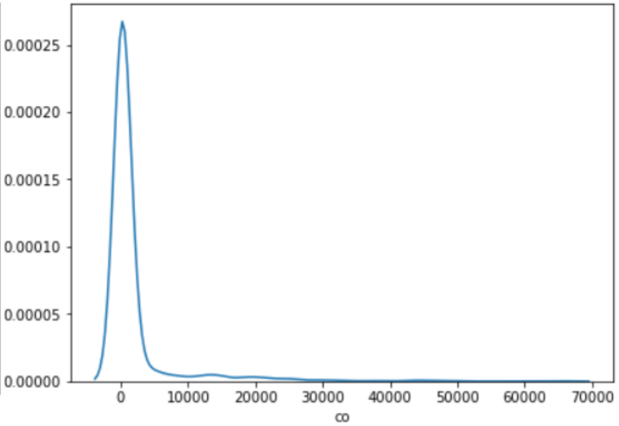
$$z = \frac{x - \mu}{\sigma}$$

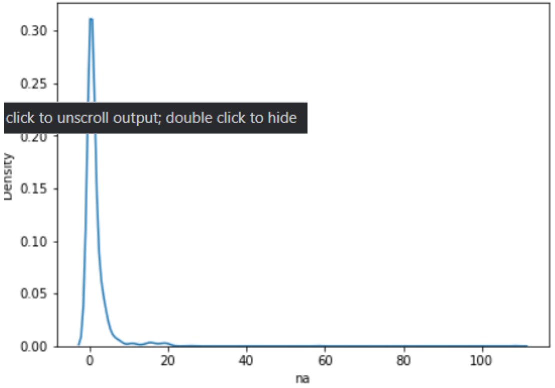
Data points with z-scores that fall above threshold value are potential outliers

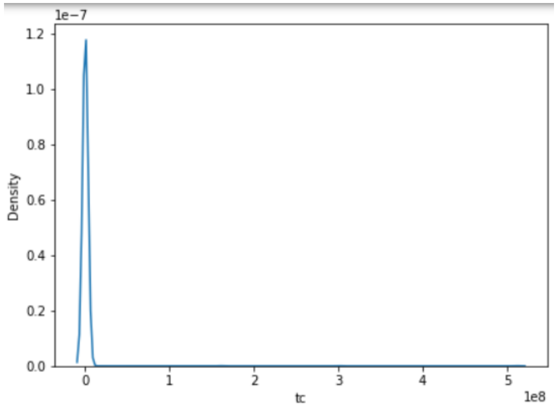
Distribution of each attribute

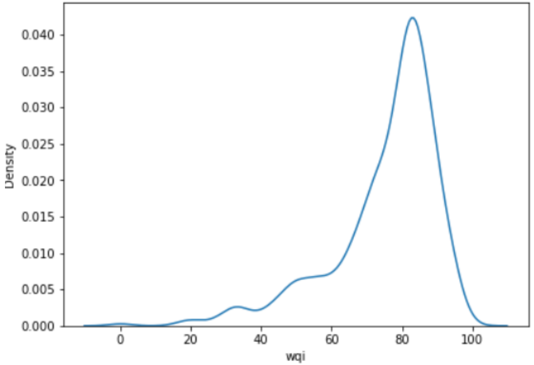


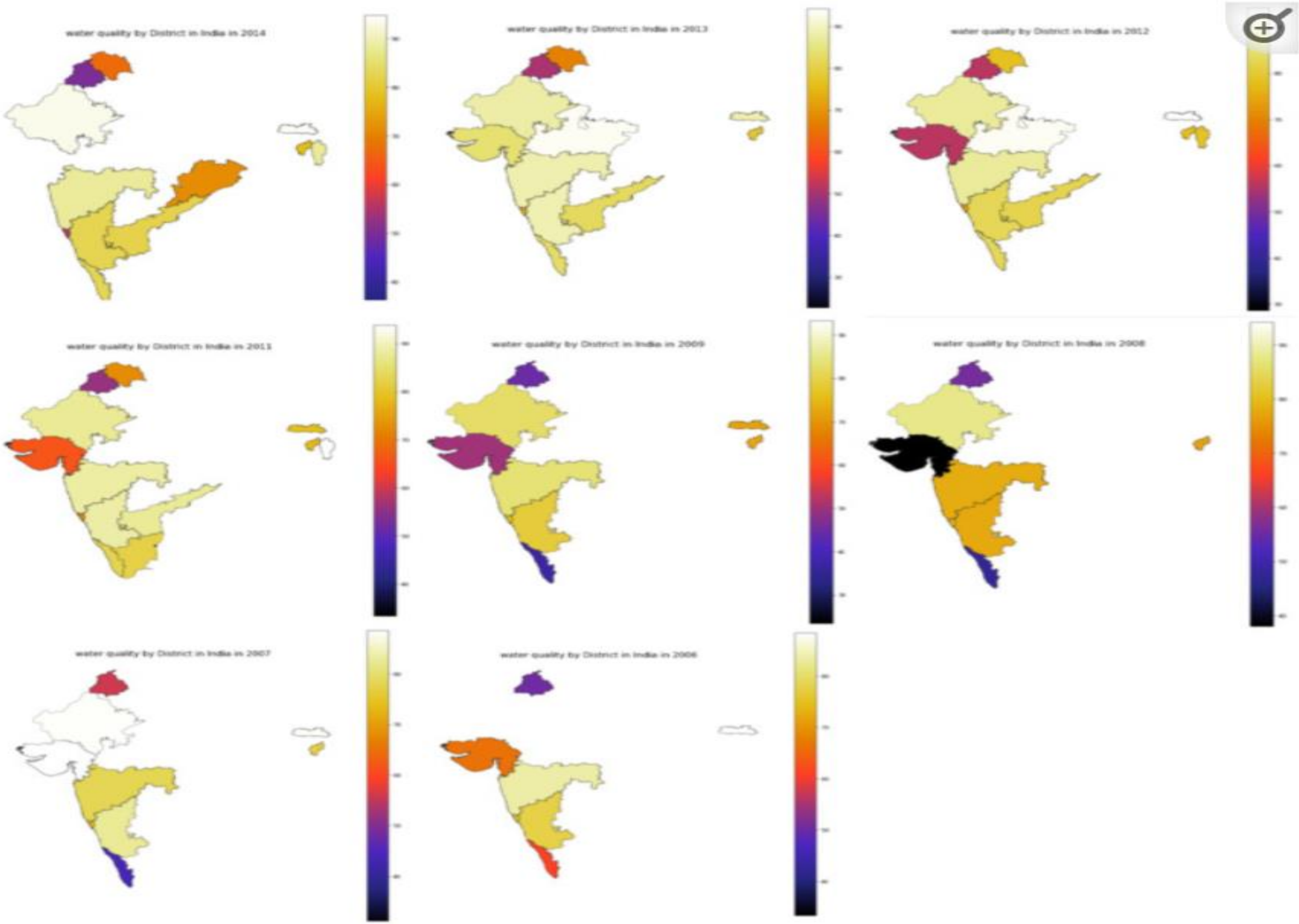











water quality by District in India in 2014
water quality by District in India in 2013
water quality by District in India in 2012
water quality by District in India in 2011
water quality by District in India in 2009
water quality by District in India in 2008
water quality by District in India in 2007
water quality by District in India in 2006

```python
#Applying Gradient Boosting
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.datasets import make_regression
gb = GradientBoostingRegressor(n_estimators=30, learning_rate=0.1, max_depth=6
gb.fit(X_train_final, y_train_final)
y_predict = gb.predict(X_test_final)
```

R2 = 0.897322314053188

R2 = 0.7602114513363876

```python
from sklearn.tree import DecisionTreeRegressor
Treereg = DecisionTreeRegressor()
Treereg.fit(X_train_final,y_train_final)
y_pred = Treereg.predict(X_test_final)
```

```python
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor()
reg.fit(X_train_final,y_train_final)
y_pred = reg.predict(X_test_final)
```

R2 = 0.882445986645837

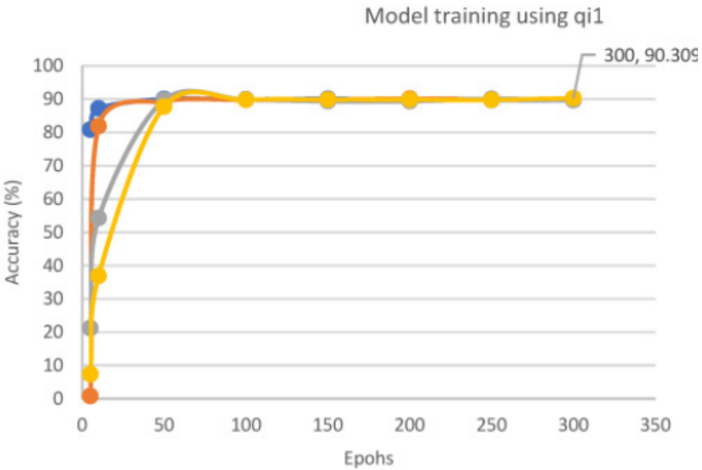
```python
print(mse)
print(r2)
```

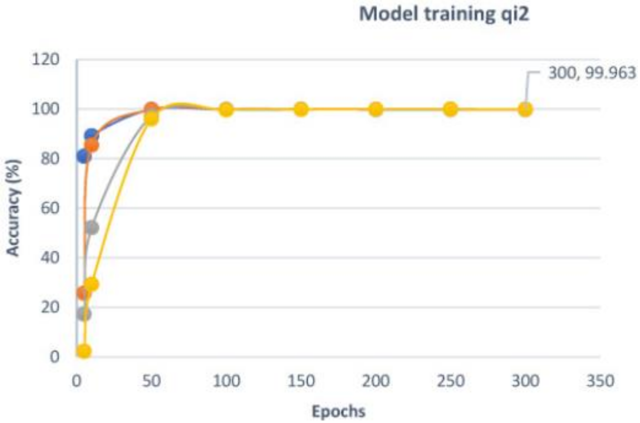
```
1.6635807910485109e-28
1.0
```

```python
#Performing Linear Regression
model = LinearRegression(fit_intercept = True)
```

```python
model.fit(X_train_n, y_train_n)
y_pred = model.predict(X_test_n)
mse = mean_squared_error(y_test_n, y_pred)
```

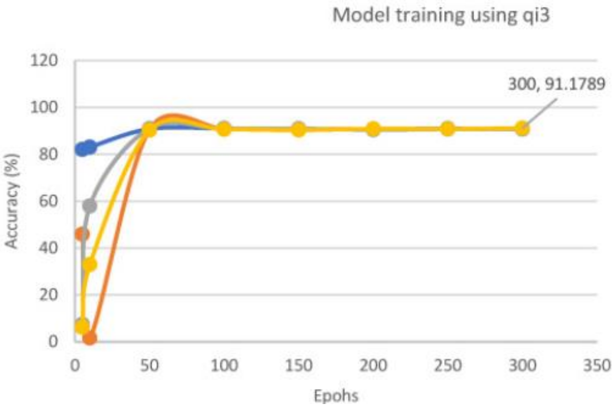
Accuracy Of Artificial Neural Network models





Qi1 = Raw Data

Qi2 = Normalized Data



Qi3 = Unit Weighted Data

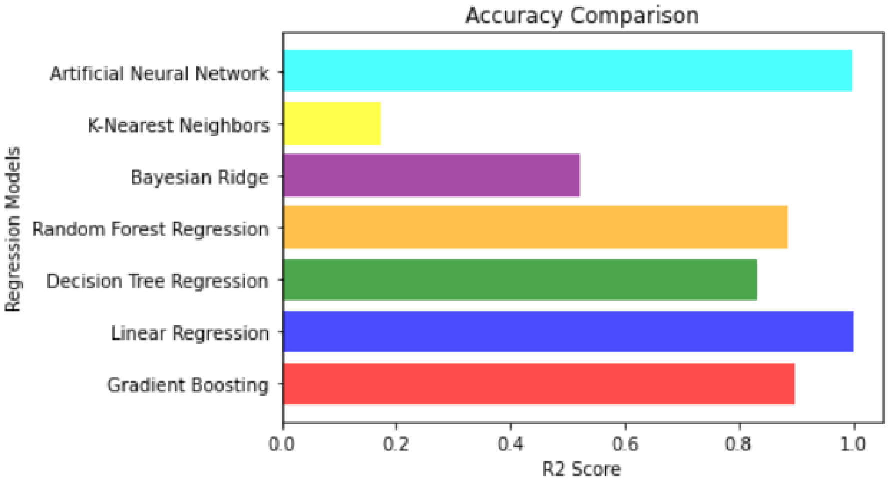
(using official sources)

FINAL RESULTS FROM ALL MODELS USED

```python
import matplotlib.pyplot as plt
labels = ['Gradient Boosting','Linear Regression','Decision Tree Regression'
values = [r2_gradb,r2_lrNorm,r2_tree,r2_forest,r2_br,r2_knn,r2_annNorm]
#colors = ['red', 'blue', 'green', 'orange', 'purple', 'yellow', 'cyan']
colors = [(1, 0, 0, 0.7),  # Red
          (0, 0, 1, 0.7),  # Blue
          (0, 0.5, 0, 0.7),  # Green
          (1, 0.65, 0, 0.7),  # Orange
          (0.5, 0, 0.5, 0.7),  # Purple
          (1, 1, 0, 0.7),  # Yellow
          (0, 1, 1, 0.7)]  # Cyan

plt.barh(labels, values,color=colors)
plt.xlabel('R2 Score')
plt.ylabel('Regression Models')
plt.title('Accuracy Comparison')

plt.show()
```



Thank You!

Harshvardhan Jouhary (2020B2A31623P)
Aryaman Chauhan (2020B5A72006P)
Devansh (2020B5A72001P)