1 Report on Spillway opening month prediction app

This app is made to predict when the spillway open.

1.1 Data collection

To read the excel data pandas library is used in python

Data collecting

```
data = pd.read_excel("spillway.xlsx")
```

Python

1.2 Data Visualization

Data visualization

```
data.head(10)
```

Python

	MONTH	DATE	MAX	MIN	CURRENT	GROSS	DEAD	AVAILABLE	Spillway Discharge	BOTTOM OUTLET	...	SPILLWAY SILL LEVEL	Sleeve Valve Discharge (MCM)	Unnamed: 19	HMIS	Inflow cumulative	Unnamed: 22	P Cumul (M
0	December	2014-12-29	438.0	370	438.09	722.0	34.0	NaN	NaN	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
1	December	2014-12-30	438.0	370	438.02	722.0	34.0	NaN	NaN	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
2	December	2014-12-31	438.0	370	438.00	722.0	34.0	NaN	NaN	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
3	January	2015-01-01	438.0	370	438.01	722.0	34.0	NaN	9.76	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
4	January	2015-01-02	438.0	370	437.98	722.0	34.0	NaN	6.06	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
5	January	2015-01-03	438.0	370	437.98	722.0	34.0	NaN	2.34	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
6	January	2015-01-04	438.0	370	438.00	722.0	34.0	NaN	2.73	NaN	...	430.0	NaN	NaN	NaN	NaN	NaN	
7	January	2015-	438.0	370	437.98	722.0	34.0	NaN	3.03	NaN		430.0	NaN	NaN	NaN	NaN	NaN	

1.3 Data Preprocessing

- Checked for the null values

Null values

```
data.isnull().sum()
```

```
MONTH                                    2823
DATE                                        0
MAX                                         0
MIN                                         0
CURRENT                                     6
GROSS                                      59
DEAD                                      537
AVAILABLE                                 584
Spillway Discharge                       2149
BOTTOM OUTLET                            2476
Sleeve Valve                             2301
POWER                                      31
TOTAL ENERGY GENERATED (GWh)               31
TOTAL OUT FLOW (MCM)                       29
INFLOW \n(MCM)                            654
RAIN FALL\nmm                             988
AVAILABLE \nPERCENTAGE ON\nACTIVE  STORAGE    1318
SPILLWAY SILL LEVEL                       235
Sleeve Valve Discharge (MCM)             3103
Unnamed: 19                              2730
HMIS                                     1633
```

- Deleted unwanted columns

```
columns_to_delete = ['MONTH', 'MAX', 'MIN', 'AVAILABLE', 'Unnamed: 19', 'Unnamed: 25', 'Unnamed: 26', 'Unnamed: 22', 'POWER',
       'TOTAL ENERGY GENERATED (GWh)', 'AVAILABLE', 'AVAILABLE \nPERCENTAGE ON\nACTIVE  STORAGE', 'Inflow cumulative', 'Power Cumulative (MCM)',
       'Total Out flow Cumulative (MCM)']
```

```
data.drop(columns=columns_to_delete, inplace=True)
```

- If the remaining columns have null values replace it with the default values

Remove null values

```
data['Spillway Discharge'].fillna(0, inplace=True)
```

Python

```
data.head()
```

Python

	DATE	CURRENT	GROSS	DEAD	Spillway Discharge	BOTTOM OUTLET	Sleeve Valve	TOTAL OUT FLOW (MCM)	INFLOW \n(MCM)	RAIN FALL\nmm	SPILLWAY SILL LEVEL	Sleeve Valve Discharge (MCM)	HMIS
0	2014-12-29	438.09	722.0	34.0	0.00	NaN	NaN	0.00	NaN	NaN	430.0	NaN	NaN
1	2014-12-30	438.02	722.0	34.0	0.00	NaN	NaN	0.00	NaN	NaN	430.0	NaN	NaN
2	2014-12-31	438.00	722.0	34.0	0.00	NaN	NaN	0.00	NaN	NaN	430.0	NaN	NaN
3	2015-01-01	438.01	722.0	34.0	9.76	NaN	NaN	21.63	NaN	0	430.0	NaN	NaN
4	2015-01-02	437.98	722.0	34.0	6.06	NaN	NaN	17.21	NaN	0	430.0	NaN	NaN

```
data['BOTTOM OUTLET'].fillna(0, inplace=True)

data['Sleeve Valve'].fillna(0, inplace=True)
```

```
data['INFLOW \n(MCM)'].fillna(0, inplace=True)

data['RAIN FALL\nmm'].fillna(0, inplace=True)

data['Sleeve Valve Discharge (MCM)'].fillna(0, inplace=True)

data['HMIS'].fillna(0, inplace=True)
```

- Drop if any remaining columns have null values

```
data.dropna(inplace=True)

data.isnull().sum()
```

```
DATE                          0
CURRENT                       0
GROSS                         0
DEAD                          0
Spillway Discharge            0
BOTTOM OUTLET                 0
Sleeve Valve                  0
TOTAL OUT FLOW (MCM)          0
INFLOW \n(MCM)                0
RAIN FALL\nmm                 0
SPILLWAY SILL LEVEL           0
Sleeve Valve Discharge (MCM)  0
HMIS                          0
dtype: int64
```

- Split data into day, month, year

```python
date_column = 'DATE'  # Replace with the actual column name

# Convert the date column to a datetime object
data[date_column] = pd.to_datetime(data[date_column])

# Extract day, month, and year using the dt accessor
data['Day'] = data[date_column].dt.day
data['Month'] = data[date_column].dt.month
data['Year'] = data[date_column].dt.year
```

Python

```python
data
```

Python

	DATE	CURRENT	GROSS	DEAD	Spillway Discharge	BOTTOM OUTLET	Sleeve Valve	TOTAL OUT FLOW (MCM)	INFLOW \n(MCM)	RAIN FALL\nmm	SPILLWAY SILL LEVEL	Sleeve Valve Discharge (MCM)	HMIS	Day	Month	Year
0	2014-12-29	438.09	722.0	34.0	0.00	0.0	0.000	0.000	0.000	0	430.0	0.0	0	29	12	2014
1	2014-12-30	438.02	722.0	34.0	0.00	0.0	0.000	0.000	0.000	0	430.0	0.0	0	30	12	2014
2	2014-	438.00	722.0	34.0	0.00	0.0	0.000	0.000	0.000	0	430.0	0.0	0	31	12	2014

- Store modified data into a pickle file for later access

```python
import pickle
```

```python
pickle.dump(data,open('data_without_cluster.pickle','wb'))
```

- Assign Feature (data given for prediction) and target (data need to predict) columns

```python
y_data = data["Spillway Discharge"]
```

```python
x_data = data['CURRENT']
```

+ Code + Markdown

```python
y_data
```

```python
x_data
```

```python
X = x_data.values.reshape(-1, 1)
```

```python
y = y_data.values.reshape(-1, 1)
```

- Split x (features), y (targets) into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

- Fit the model (training data set) to a logistic regression model which can be used to find the equation between x and y.

```
model = LinearRegression()
```

- Calculate and analyze the accuracy of the model by comparing the predicted value with testing values.

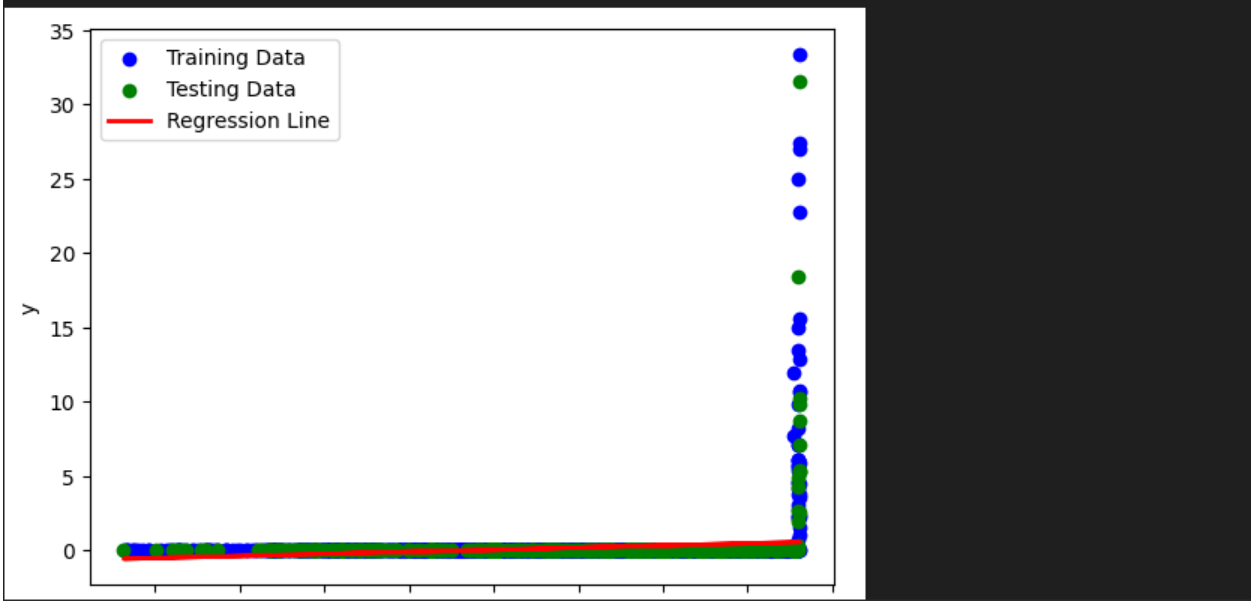
```
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'Mean Absolute Error: {mae}')
print(f'R-squared: {r2}')


import matplotlib.pyplot as plt
```

- Visualizing the values of predicted and actual values

```
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.scatter(X_test, y_test, color='green', label='Testing Data')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

✓ 0.2s



- Load the modified data in the pickle file

```
data = pickle.load(open('data_without_cluster.pickle',mode='rb'))
```

✓ 0.4s

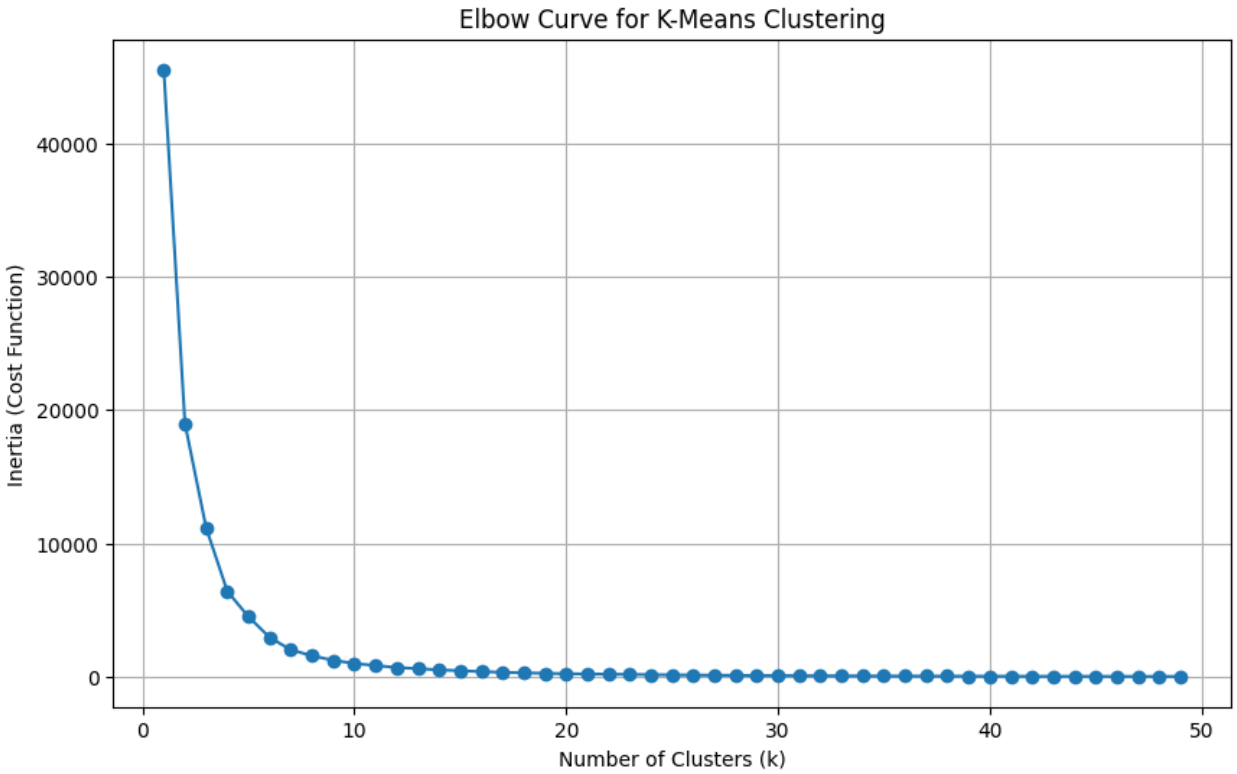
- Create a function to take month and year as an array input

```
def month_year(x, y):
    result_array = np.array([[str(i), x, y] for i in range(1, 31)])
    result_array_reshaped = result_array.reshape(-1, 3)
    return result_array_reshaped
```

✓ 0.0s

- Using k mean algorithm to predict number of clusters

```python
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Assuming data and X are already defined as in your previous code

# Define a range of k values
k_values = range(1, 50)  # You can adjust the range as needed

# Initialize an empty list to store inertia values
inertia_values = []

# Iterate through different k values
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(data["TOTAL OUT FLOW (MCM)"].values.reshape(-1, 1))
    # Append the inertia value to the list
    inertia_values.append(kmeans.inertia_)

# Plotting the Elbow Curve
plt.figure(figsize=(10, 6))
plt.plot(k_values, inertia_values, marker='o')
plt.title('Elbow Curve for K-Means Clustering')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia (Cost Function)')
plt.grid(True)
plt.show()
```



Elbow starts to bend at 6, therefore 6 clusters

- Substituting k=6

```
X = data['Spillway Discharge'].values.reshape(-1, 1)

# Choose the number of clusters (k)
k = 6
# Apply K-Means clustering
kmeans = KMeans(n_clusters=k, random_state=42)
data['Cluster'] = kmeans.fit_predict(X)
```

```
data
```

Python

	CURRENT	GROSS	DEAD	Spillway Discharge	BOTTOM OUTLET	Sleeve Valve	TOTAL OUT FLOW (MCM)	INFLOW \n(MCM)	RAIN FALL\nmm	SPILLWAY SILL LEVEL	Sleeve Valve Discharge (MCM)	HMIS	Day	Month	Year	Cluster
0	438.09	722.0	34.0	0.00	0.0	0.000	0.000	0.000	0	430.0	0.0	0	29	12	2014	0
1	438.02	722.0	34.0	0.00	0.0	0.000	0.000	0.000	0	430.0	0.0	0	30	12	2014	0
2	438.00	722.0	34.0	0.00	0.0	0.000	0.000	0.000	0	430.0	0.0	0	31	12	2014	0
3	438.01	722.0	34.0	9.76	0.0	0.000	21.630	0.000	0	430.0	0.0	0	1	1	2015	2
4	437.98	722.0	34.0	6.06	0.0	0.000	17.210	0.000	0	430.0	0.0	0	2	1	2015	4

- Save the data as a pickle file

```
pickle.dump(data,open('preprocessed_data.pickle','wb'))
```

- Predict the gate opening level using Random forest algorithm

```
X = data[['Day', 'Month', 'Year']]
y = data['Cluster']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier
andomForestClassifier(random_state=42)
```

- Analyzing the accuracy report

```
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Classification Report:\n", report)
```

```
Accuracy: 0.9919517102615694
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       493
           2       1.00      1.00      1.00         1
           3       0.00      0.00      0.00         0
           4       0.33      0.50      0.40         2
           5       1.00      1.00      1.00         1

    accuracy                           0.99       497
   macro avg       0.67      0.70      0.68       497
weighted avg       1.00      0.99      0.99       497
```

Ended with **99%** accuracy.

- Developing an application

```python
class PredictionApp:
    def __init__(self, master):
        self.master = master
        self.master.title("Prediction App")

        # Create and set up widgets
        self.label_month = ttk.Label(master, text="Month:")
        self.entry_month = ttk.Entry(master)
        self.label_year = ttk.Label(master, text="Year:")
        self.entry_year = ttk.Entry(master)
        self.predict_button = ttk.Button(master, text="Predict", command=self.make_prediction)

        # Layout
        self.label_month.grid(row=0, column=0, padx=10, pady=10)
        self.entry_month.grid(row=0, column=1, padx=10, pady=10)
        self.label_year.grid(row=1, column=0, padx=10, pady=10)
        self.entry_year.grid(row=1, column=1, padx=10, pady=10)
        self.predict_button.grid(row=2, column=0, columnspan=2, pady=10)
```

```python
    def make_prediction(self):
        try:
            month_value = int(self.entry_month.get())
            year_value = int(self.entry_year.get())
        except ValueError:
            messagebox.showerror("Error", "Please enter valid numeric values for Month and Year.")
            return

        month_data = month_year(month_value, year_value)

        try:
            prediction = clf.predict(month_data)

            tolerance = 1e-6  # You can adjust this based on your specific case

            if prediction is not None and np.any(np.abs(prediction) > tolerance):
                prediction_result = "open"
            else:
                prediction_result = "close"
            messagebox.showinfo("Prediction", f"The spillway will {prediction_result}")
        except Exception as e:
            messagebox.showerror("Error", f"An error occurre {str(e)}")
```

```python
if __name__ == "__main__":
    # Create and train your model (Assuming data is defined)
    X = data[['Day', 'Month', 'Year']]
    y = data['Cluster']
    X_train, _, y_train, _ = train_test_split(X, y, test_size=0.2)
    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)

    # Create the main application window
    root = tk.Tk()
    app = PredictionApp(root)
    root.mainloop()
```

- Final output