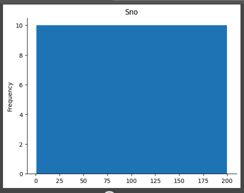
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import seaborn as sns
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import mean_absolute_error, r2_score
```

```
data=pd.read_csv('/content/Final_dataset.csv')
```

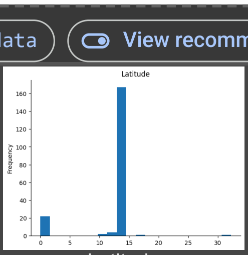
```
data.head()
```

	Sno	Ward Name	Latitude	Longitude	Distance from Adyar River(km)	Distance from Cooum River(km)	Distance from Kosasthalaiyar River(km)	Elevation	Rainfall (in cm)	Wat Level(fee
0	1	KATHIVAKKAM	13.216133	80.318177	33.72	55.67	66.25	16	24.09	9.
1	2	ENNORE	13.214600	80.320300	62.88	19.21	32.94	12	30.85	8.
2	3	ERNAVOOR	13.189600	80.303900	15.91	62.97	14.27	20	23.11	10.
3	4	AJAX	13.172100	80.305100	59.61	40.64	23.18	13	16.37	5.
4	5	TIRUVOTTRIYUR	13.164300	80.300100	13.82	61.12	20.09	16	16.62	8.

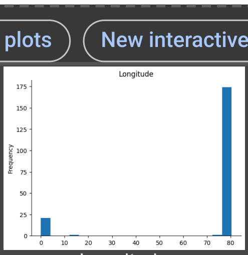
Next steps: Generate code with data View recommended plots New interactive sheet



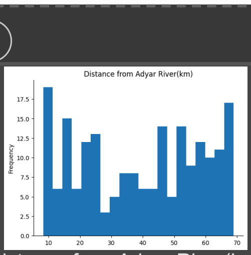
Sno



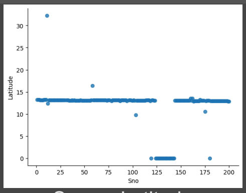
Latitude



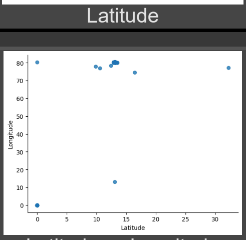
Longitude



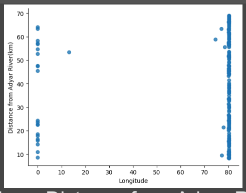
Distance from Adyar River(km)



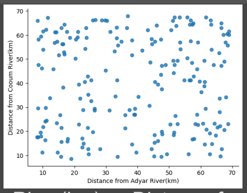
Sno vs Latitude



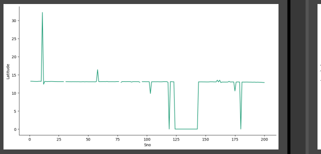
Latitude vs Longitude



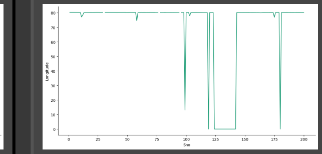
Longitude vs Distance from Adyar River(km)



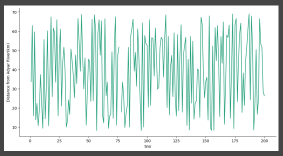
Distance from Adyar River(km) vs Distance from Cooum River(km)



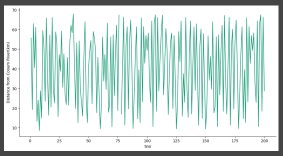
Sno vs Latitude



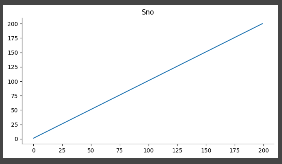
Sno vs Longitude



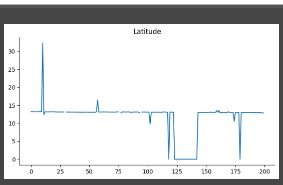
Sno vs Distance from Adyar River(km)



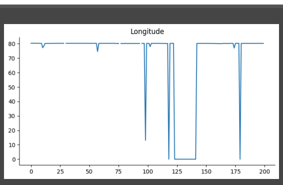
Sno vs Distance from Coooum River(km)



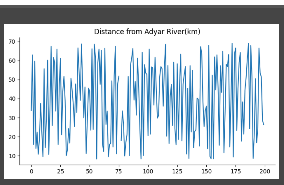
Sno



Latitude



Longitude



Distance from Adyar River(km)

```
data.columns
```

```
Index(['Sno', ' Ward Name', 'Latitude', 'Longitude',
       'Distance from Adyar River(km)', 'Distance from Cooum River(km)',
       'Distance from Kosasthalaiyar River(km)', 'Elevation',
       'Rainfall (in cm)', 'Water Level(feet)'],
      dtype='object')
```

```
data.describe()
```

	Sno	Latitude	Longitude	Distance from Adyar River(km)	Distance from Cooum River(km)	Distance from Kosasthalaiyar River(km)	Elevation	Rainfall (in cm)	Wate Level(feet
count	200.000000	197.000000	197.000000	199.000000	199.000000	199.000000	200.000000	200.000000	200.00000
mean	100.500000	11.695948	71.249964	39.207487	40.060804	41.212965	15.735000	21.474150	9.08905
std	57.879185	4.393748	25.138027	19.445838	18.939417	18.514302	5.125047	5.230115	3.50854
min	1.000000	0.000000	0.000000	8.270000	8.560000	8.340000	7.000000	12.570000	2.00000
25%	50.750000	12.984200	80.164800	21.355000	21.960000	23.700000	11.000000	16.867500	6.00000
50%	100.500000	13.073200	80.218000	40.280000	41.260000	41.720000	15.000000	20.910000	9.00000
75%	150.250000	13.110300	80.256200	57.020000	58.020000	58.400000	20.000000	25.880000	12.00000
max	200.000000	32.243200	80.320300	68.970000	67.920000	69.830000	25.000000	30.890000	15.00000

DATA PREPROCESSING

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

	0
Sno	0
Ward Name	0
Latitude	3
Longitude	3
Distance from Adyar River(km)	1
Distance from Cooum River(km)	1
Distance from Kosasthalaiyar River(km)	1
Elevation	0
Rainfall (in cm)	0
Water Level(feet)	0

dtype: int64

```
data.drop(data.index[(data[" Ward Name"] == "4TH MAIN RD")],axis=0,inplace=True)
```

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

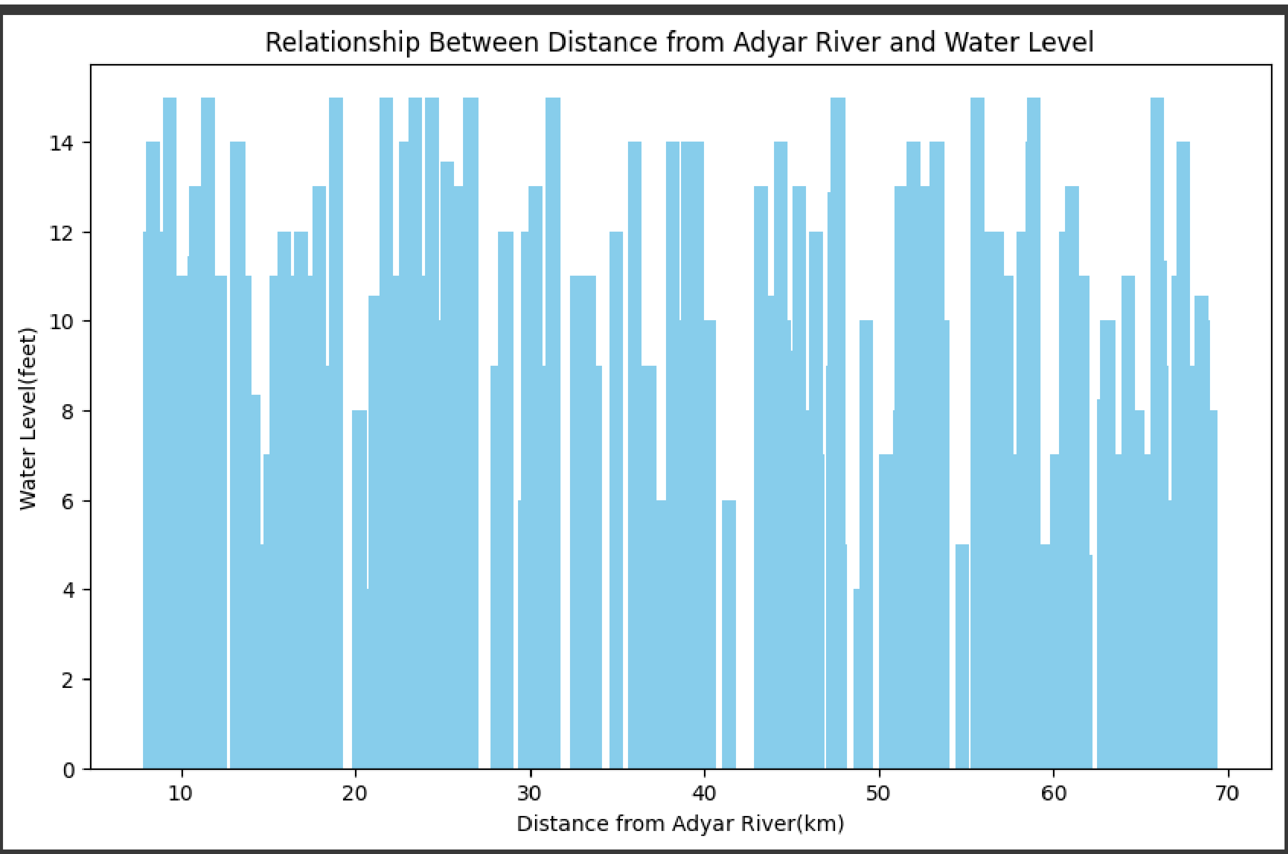
```
                                              0
                                 Sno          0
                           Ward Name          0
                            Latitude          3
                           Longitude          3
         Distance from Adyar River(km)        1
         Distance from Cooum River(km)        1
  Distance from Kosasthalaiyar River(km)      1
                           Elevation          0
                     Rainfall (in cm)         0
                    Water Level(feet)         0

dtype: int64
```

```
data.dropna(inplace=True)
```

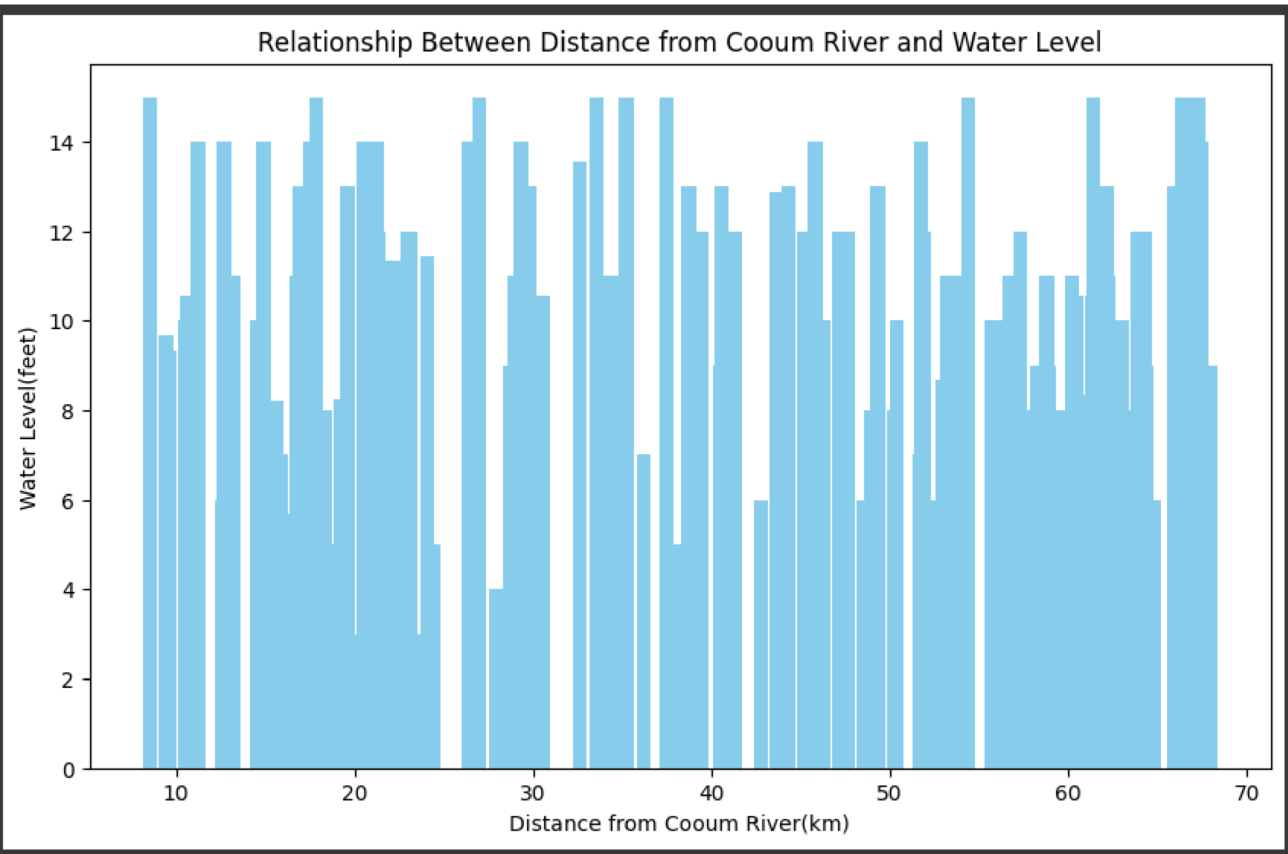
```
data.isnull().sum()
```

```
                                              0
                                 Sno          0
                           Ward Name          0
                            Latitude          0
                           Longitude          0
         Distance from Adyar River(km)        0
         Distance from Cooum River(km)        0
  Distance from Kosasthalaiyar River(km)      0
                           Elevation          0
                     Rainfall (in cm)         0
                    Water Level(feet)         0

dtype: int64
```

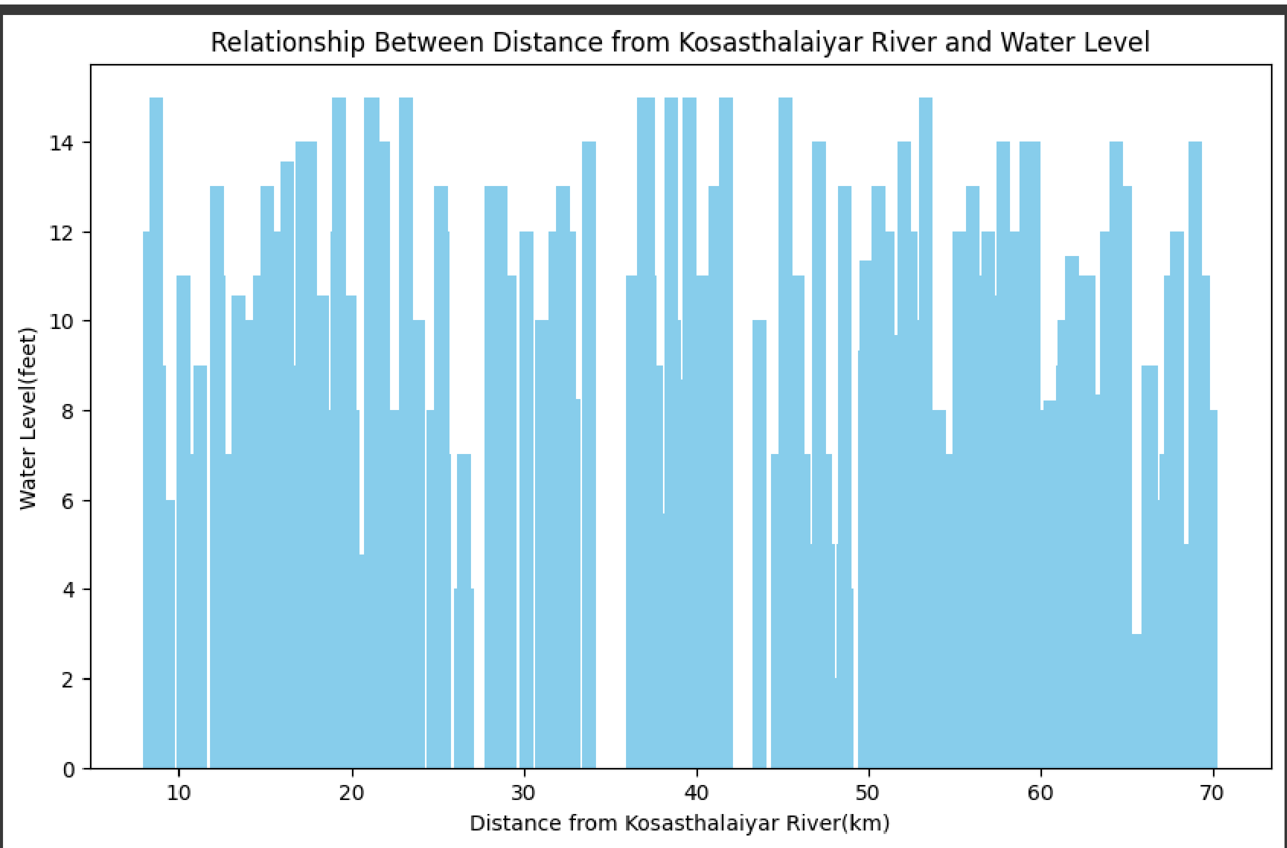
```
import matplotlib.pyplot as plt

# Extracting the feature values from your dataset
distances = data['Distance from Adyar River(km)']
water_levels = data['Water Level(feet)']

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(distances, water_levels, color='skyblue')

# Adding labels and title
plt.xlabel('Distance from Adyar River(km)')
plt.ylabel('Water Level(feet)')
plt.title('Relationship Between Distance from Adyar River and Water Level')

# Show the plot
plt.show()
```



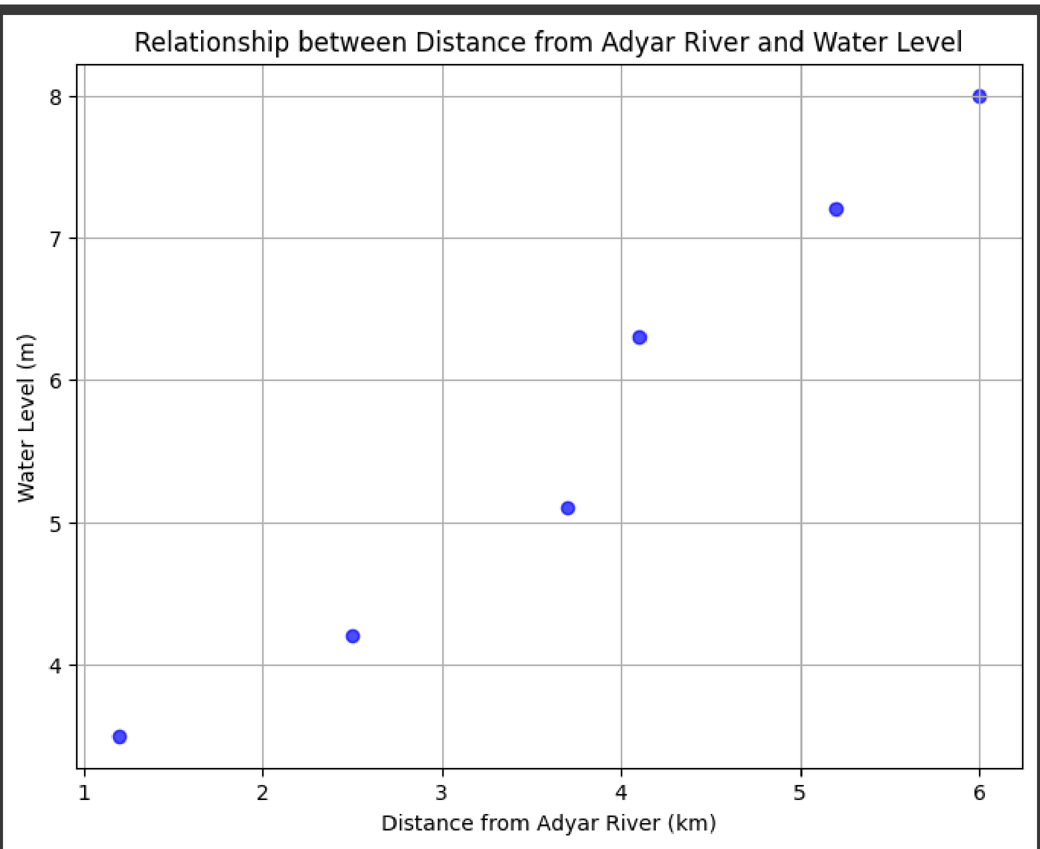
```
import matplotlib.pyplot as plt

# Extracting the feature values from your dataset
distances = data['Distance from Cooum River(km)']
water_levels = data['Water Level(feet)']

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(distances, water_levels, color='skyblue')

# Adding labels and title
plt.xlabel('Distance from Cooum River(km)')
plt.ylabel('Water Level(feet)')
plt.title('Relationship Between Distance from Cooum River and Water Level')

# Show the plot
plt.show()
```



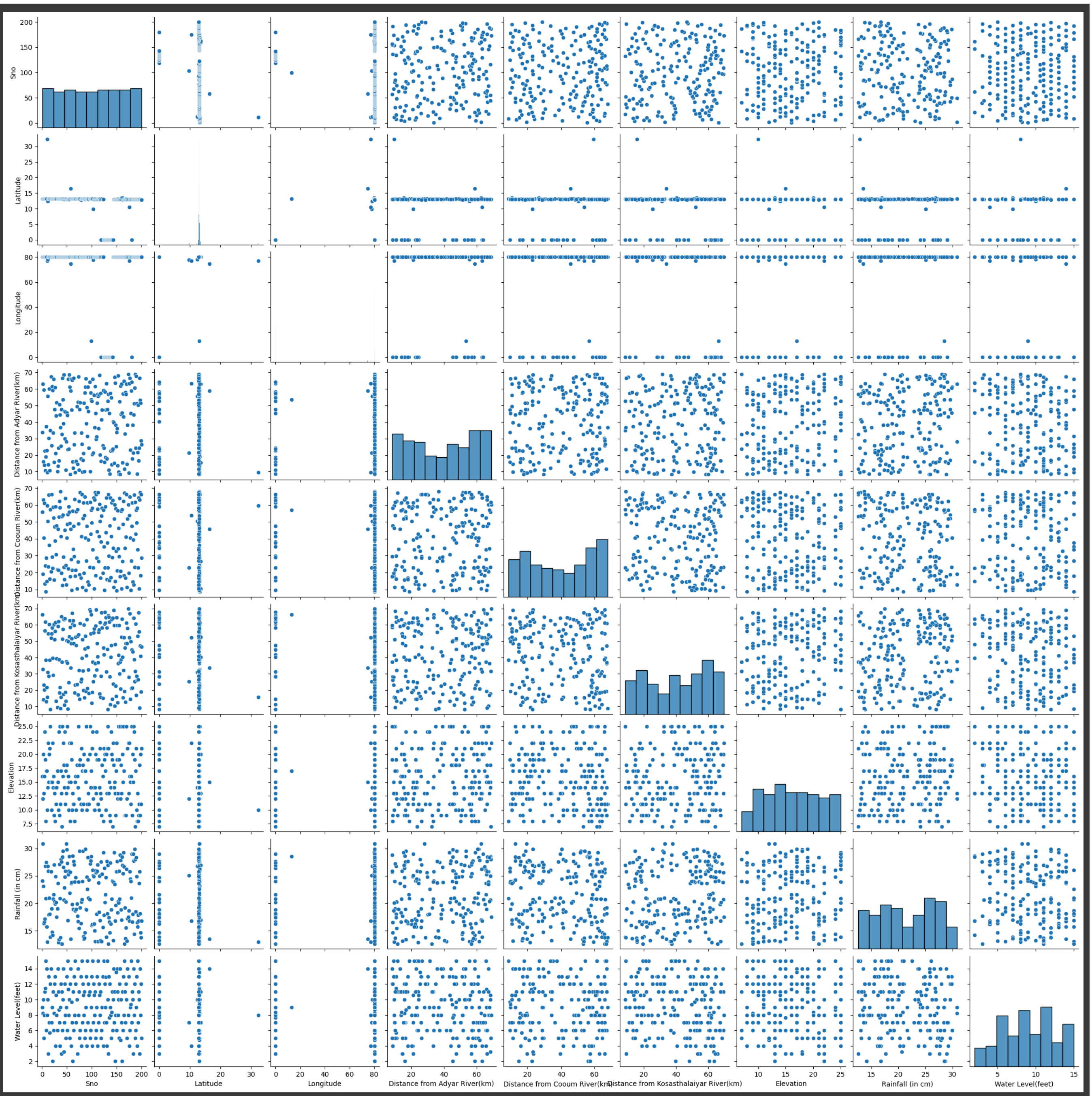
```
import matplotlib.pyplot as plt

# Extracting the feature values from your dataset
distances = data['Distance from Kosasthalaiyar River(km)']
water_levels = data['Water Level(feet)']

# Plotting the bar graph
plt.figure(figsize=(10, 6))
plt.bar(distances, water_levels, color='skyblue')

# Adding labels and title
plt.xlabel('Distance from Kosasthalaiyar River(km)')
plt.ylabel('Water Level(feet)')
plt.title('Relationship Between Distance from Kosasthalaiyar River and Water Level')

# Show the plot
plt.show()
```



```
import matplotlib.pyplot as plt

# Sample data for Distance from Adyar River and Water Level
distance_from_adyar_river = [1.2, 2.5, 3.7, 4.1, 5.2, 6.0]  # Example distance values in kilometers
water_level = [3.5, 4.2, 5.1, 6.3, 7.2, 8.0]  # Example water level values in meters

# Create a scatter plot for Distance from Adyar River vs. Water Level
plt.figure(figsize=(8, 6))
plt.scatter(distance_from_adyar_river, water_level, color='blue', alpha=0.7)
plt.title('Relationship between Distance from Adyar River and Water Level')
plt.xlabel('Distance from Adyar River (km)')
plt.ylabel('Water Level (m)')
plt.grid(True)
plt.show()
```



```
sns.pairplot(data)
plt.show()
```



```
X = data[['Distance from Adyar River(km)', 'Distance from Cooum River(km)', 'Distance from Kosasthalaiyar River(km)',
y = data['Water Level(feet)']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
tree_model = DecisionTreeRegressor(random_state=42)
```

```
tree_model.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor ⓘ ?
DecisionTreeRegressor(random_state=42)
```

```
forest_model = RandomForestRegressor(random_state=42)
```

```
forest_model.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
boosting_model = GradientBoostingRegressor(random_state=42)
```

```
boosting_model.fit(X_train, y_train)
```

```
▾ GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

```
tree_preds = tree_model.predict(X_test)
forest_preds = forest_model.predict(X_test)
boosting_preds = boosting_model.predict(X_test)
```

```
mae = mean_absolute_error(y_test, tree_preds)
print("\nMean Absolute Error (MAE) (Decision Tree):", mae)
mse = mean_squared_error(y_test, tree_preds)
print("Mean Squared Error (Decision Tree):", mse)
```

```

Mean Absolute Error (MAE) (Decision Tree): 4.22475
Mean Squared Error (Decision Tree): 26.9159225
```

```
mae = mean_absolute_error(y_test, forest_preds)
print("\nMean Absolute Error (MAE) of Random Forest:", mae)
mse = mean_squared_error(y_test, forest_preds)
print("Mean Squared Error (Meta-Model) of Random Forest:", mse)
```

```

Mean Absolute Error (MAE) of Random Forest: 3.5914975
Mean Squared Error (Meta-Model) of Random Forest: 16.770501782750003
```

```
mae = mean_absolute_error(y_test, boosting_preds)
print("\nMean Absolute Error (MAE) of Gradient boosting :", mae)
mse = mean_squared_error(y_test, boosting_preds)
print("Mean Squared Error of Gradient Boosting:", mse)
```

```

Mean Absolute Error (MAE) of Gradient boosting : 3.663811402401471
Mean Squared Error of Gradient Boosting: 18.085988269097335
```

```
meta_X = np.column_stack((tree_preds, forest_preds, boosting_preds))
meta_model = LinearRegression()
meta_model.fit(meta_X, y_test)
```

```
▾ LinearRegression
LinearRegression()
```

```
meta_features_train = np.column_stack((tree_model.predict(X_train),
                                       forest_model.predict(X_train),
                                       boosting_model.predict(X_train)))
```

```
meta_model.fit(meta_features_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
meta_features_test = np.column_stack((tree_preds, forest_preds, boosting_preds))
meta_predictions = meta_model.predict(meta_features_test)
```

```
mae = mean_absolute_error(y_test, meta_predictions)
print("\nMean Absolute Error (MAE):", mae)
mse = mean_squared_error(y_test, meta_predictions)
print("Mean Squared Error (Meta-Model):", mse)
```

```
Mean Absolute Error (MAE): 4.22475
Mean Squared Error (Meta-Model): 26.9159225
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# Define the neural network architecture
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.2),  # Optional: Regularization with dropout
    Dense(32, activation='relu'),
    Dense(1)  # Output layer (1 neuron for regression)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)
```

```
Epoch 1/50
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 81ms/step - loss: 55.9420 - mae: 6.4058 - val_loss: 39.8511 - val_mae: 5.0188
Epoch 2/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 32.8178 - mae: 4.5463 - val_loss: 51.8984 - val_mae: 5.9080
Epoch 3/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 38.4263 - mae: 4.8595 - val_loss: 30.4181 - val_mae: 4.4380
Epoch 4/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 30.7144 - mae: 4.4351 - val_loss: 20.2681 - val_mae: 3.6239
Epoch 5/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 28.0649 - mae: 4.2853 - val_loss: 19.1525 - val_mae: 3.4998
Epoch 6/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - loss: 24.4654 - mae: 3.8568 - val_loss: 20.8169 - val_mae: 3.7372
Epoch 7/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 25.2506 - mae: 4.1765 - val_loss: 23.7931 - val_mae: 3.9590
Epoch 8/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 24.1700 - mae: 3.8968 - val_loss: 20.8864 - val_mae: 3.7449
Epoch 9/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 26.1053 - mae: 4.1650 - val_loss: 17.0114 - val_mae: 3.4042
Epoch 10/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 18.6819 - mae: 3.4507 - val_loss: 15.9267 - val_mae: 3.2581
Epoch 11/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 17.7777 - mae: 3.5053 - val_loss: 15.9713 - val_mae: 3.2775
Epoch 12/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 26.5389 - mae: 4.1526 - val_loss: 16.9503 - val_mae: 3.4001
Epoch 13/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 19.8795 - mae: 3.6465 - val_loss: 16.6287 - val_mae: 3.3560
Epoch 14/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - loss: 17.9495 - mae: 3.4046 - val_loss: 15.9235 - val_mae: 3.2279
Epoch 15/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 16.6460 - mae: 3.2776 - val_loss: 15.8789 - val_mae: 3.2134
Epoch 16/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 19.9064 - mae: 3.7110 - val_loss: 16.1192 - val_mae: 3.2700
Epoch 17/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 18.2337 - mae: 3.4994 - val_loss: 16.5324 - val_mae: 3.3469
Epoch 18/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - loss: 19.7424 - mae: 3.7257 - val_loss: 16.7075 - val_mae: 3.3705
Epoch 19/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - loss: 17.0863 - mae: 3.4606 - val_loss: 16.5865 - val_mae: 3.3268
Epoch 20/50
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 18.9372 - mae: 3.6665 - val_loss: 16.8051 - val_mae: 3.3516
Epoch 21/50
```