Documentation and Reporting

Project Title: Weather Time Series Analysis and Temperature Prediction

1. Problem Statement

The goal of this project is to develop a predictive model that accurately forecasts temperature based on weather data from the dataset. By analyzing various weather features, the model aims to predict temperature with high accuracy.

2. Expected Outcome

Develop a model that accurately predicts temperature with low error rates. The expected outcome includes detailed exploratory data analysis, preprocessing steps, model training, and evaluation metrics.

3. Data Collection

Data Source: Max Planck Weather Time Series dataset (max_planck_weather_ts.csv).
Dataset:https://www.kaggle.com/datasets/arashnic/max-planck-weather-dataset?resource=download
Features: Includes T (degC), rh (%), p (mbar), VPmax (mbar), Tpot (K), Tdew (degC), VPact (mbar), VPdef (mbar), sh (g/kg), H2OC (mmol/mol), rho (g/m**3), wv (m/s), max. wv (m/s), wd (deg).
Target Variable: T (degC).

4. Data Preprocessing

Handling Missing Values

- Checked for missing values: No missing values detected.
- Dealing with Duplicate Data
- Detected and removed duplicate records:

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

Outlier Detection and Treatment

Identified outliers using box plots.

```
plt.figure(figsize=(10, 6))
sns.boxplot(data['T (degC)'])
plt.title('Box plot of Temperature (degC)')
```

```
plt.show()
```

Removed extreme outliers using the IQR method:

```
Q1 = data[['T (degC)', ...]].quantile(0.25)
Q3 = data[['T (degC)', ...]].quantile(0.75)
IQR = Q3 - Q1
filter = (data[['T (degC)', ...]] >= (Q1 - 1.5 * IQR)) & (data[['T (degC)', ...]] <= (Q3 +
1.5 * IQR))
data_filtered = data[filter.any(axis=1)]
```

Feature Scaling/Normalization

Applied StandardScaler to normalize features:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(data.drop(columns=['Date Time']))
data_scaled = pd.DataFrame(scaled_features, columns=data.columns[1:])
```

5. Exploratory Data Analysis (EDA)

- Descriptive Statistics

  Provided summary statistics:

```
print(data.describe())
```

- Visualizations

Box Plot of Temperature:

Description: Highlights outliers in temperature data.

Temperature Distribution Histogram:

```
plt.hist(data_filtered['T (degC)'], bins=30, edgecolor='k')
plt.title('Temperature Distribution')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.show()
```

Correlation Heatmap:

```
plt.figure(figsize=(12, 8))
sns.heatmap(data_filtered.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

Scatter Plot of Temperature vs. Pressure:

```
plt.figure(figsize=(10, 6))
plt.scatter(data['p (mbar)'], data['T (degC)'], alpha=0.5)
plt.title('Temperature (degC) vs. Pressure (mbar)')
plt.xlabel('Pressure (mbar)')
```

```
plt.ylabel('Temperature (degC)')
plt.show()
```

6. Feature Engineering

- Converted Date Time to datetime object:

```
data['Date Time'] = pd.to_datetime(data['Date Time'])
```

7. Data Splitting

Split the dataset into 80% training and 20% testing sets:

```
from sklearn.model_selection import train_test_split
X = data_scaled.drop(columns=['T (degC)'])
y = data_scaled['T (degC)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

8. Model Selection

Chose Linear Regression, Decision Tree, and Random Forest based on problem nature and data.

9. Model Training

Trained models using the training dataset:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

lr_model = LinearRegression()
dt_model = DecisionTreeRegressor()
rf_model = RandomForestRegressor()

lr_model.fit(X_train, y_train)
dt_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
```

10. Model Evaluation

Evaluation Metrics

Linear Regression:
lr_predictions = lr_model.predict(X_test)
lr_mse, lr_mae, lr_r2 = evaluate_model(lr_predictions, y_test)
print(f"Linear Regression - MSE: {lr_mse}, MAE: {lr_mae}, R2: {lr_r2}")

Decision Tree:
dt_predictions = dt_model.predict(X_test)
dt_mse, dt_mae, dt_r2 = evaluate_model(dt_predictions, y_test)
print(f"Decision Tree - MSE: {dt_mse}, MAE: {dt_mae}, R2: {dt_r2}")
Random Forest:

rf_predictions = rf_model.predict(X_test)
rf_mse, rf_mae, rf_r2 = evaluate_model(rf_predictions, y_test)
print(f"Random Forest - MSE: {rf_mse}, MAE: {rf_mae}, R2: {rf_r2}")

Hyperparameter Tuning

Used GridSearchCV for hyperparameter tuning of the Random Forest model:

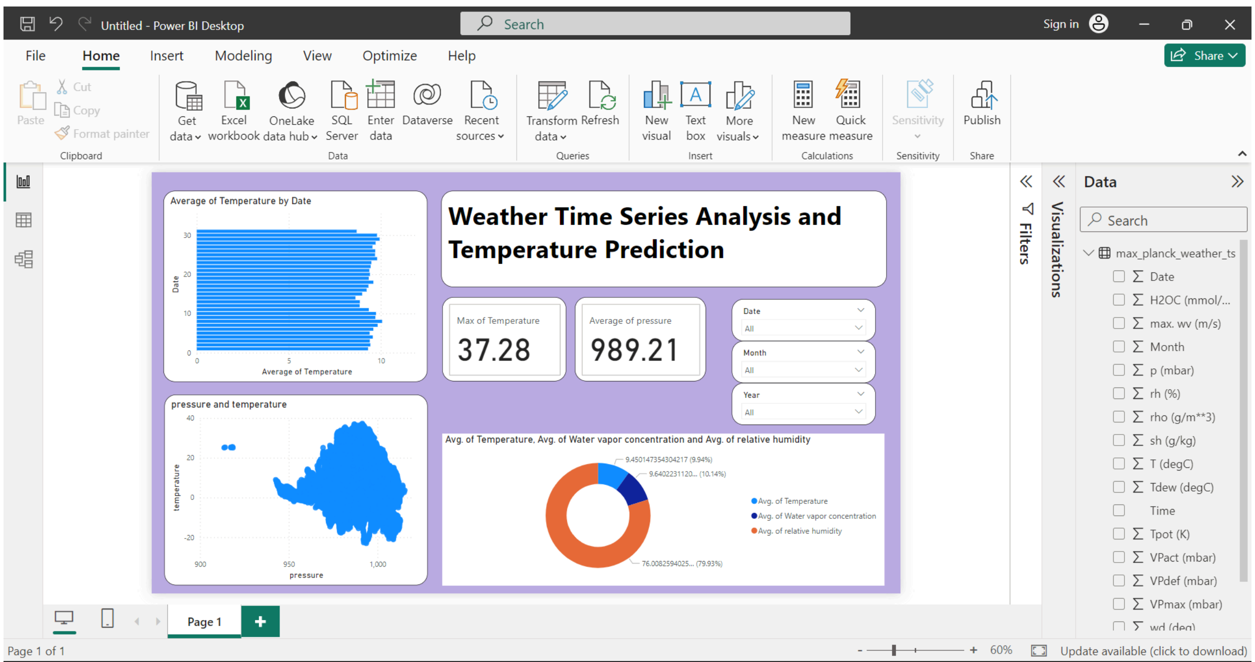
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, 30]
}
grid_search = GridSearchCV(rf_model, param_grid, cv=3,
scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_rf_model = grid_search.best_estimator_
best_rf_predictions = best_rf_model.predict(X_test)
best_rf_mse, best_rf_mae, best_rf_r2 = evaluate_model(best_rf_predictions,
y_test)
print(f"Best Random Forest - MSE: {best_rf_mse}, MAE: {best_rf_mae}, R2:
{best_rf_r2}")
```

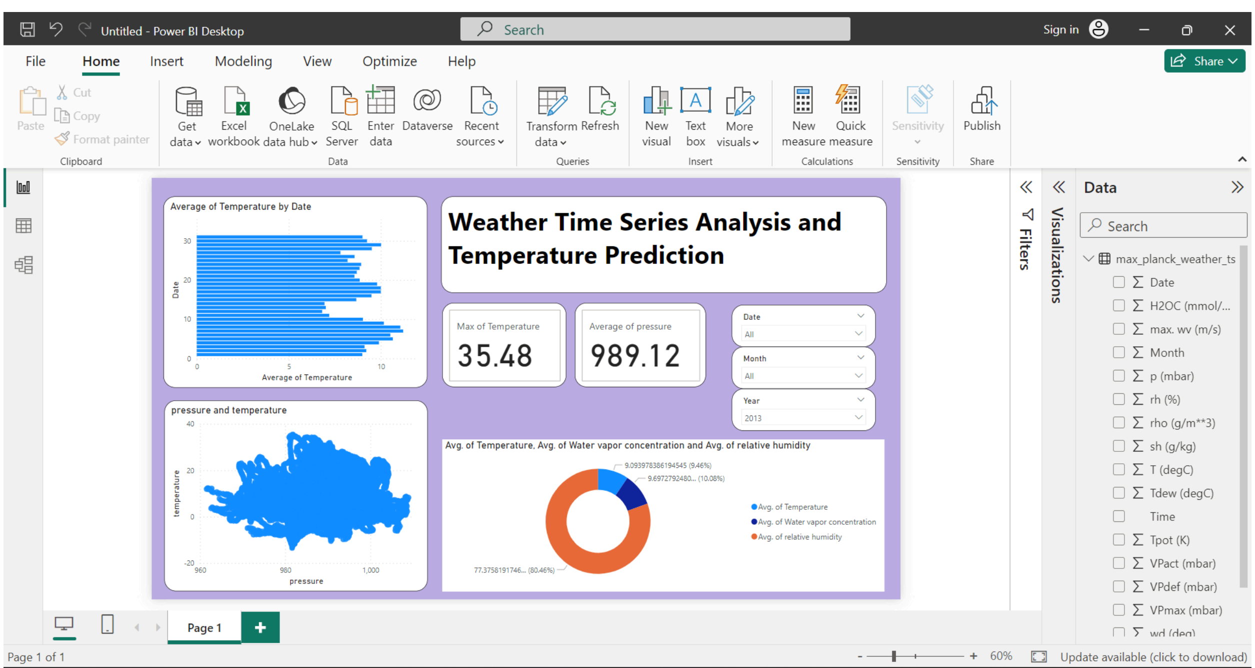
11. Model Deployment

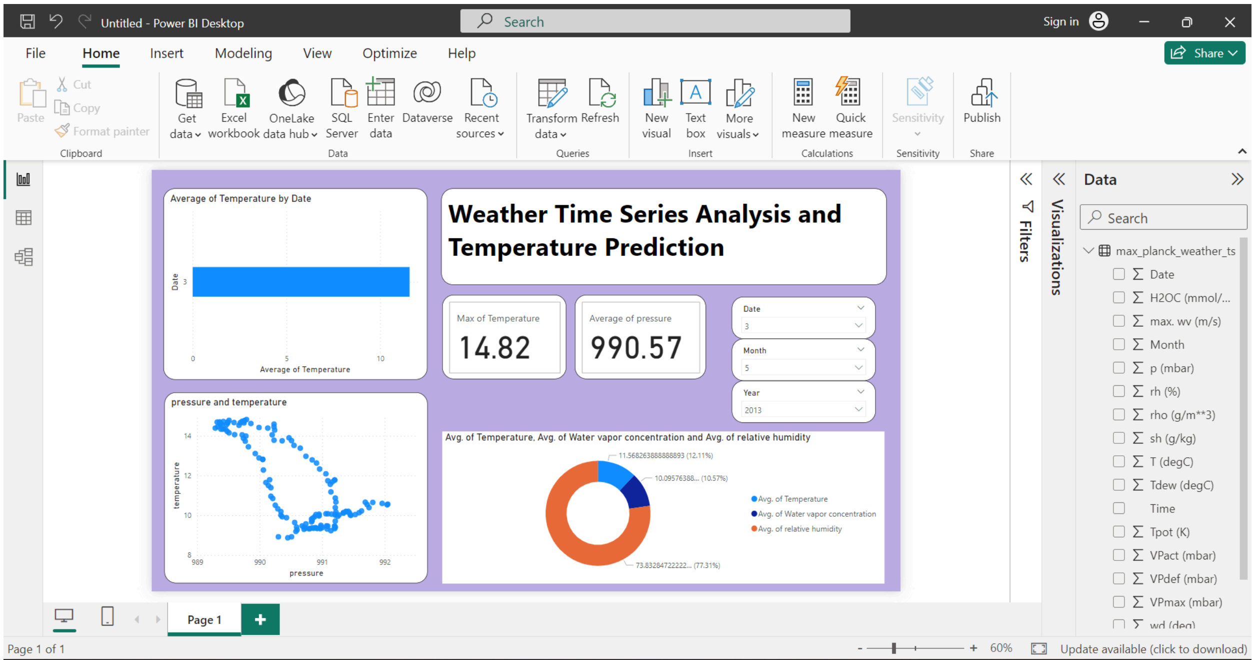
Saved the best Random Forest model using joblib:
import joblib
joblib.dump(best_rf_model, 'best_rf_model.pkl')
Loaded the saved model and made predictions:
loaded_model = joblib.load('best_rf_model.pkl')
final_predictions = loaded_model.predict(X_test)
Visualization of Forecasted Results

Compared actual vs predicted temperatures:

```
plt.figure(figsize=(14, 7))
plt.plot(data['Date Time'][-len(y_test):], y_test, label='Actual')
plt.plot(data['Date Time'][-len(y_test):], final_predictions, label='Predicted')
plt.xlabel('Date Time')
plt.ylabel('Temperature (degC)')
plt.title('Actual vs Predicted Temperature')
plt.legend()
plt.show()
```






Untitled - Power BI Desktop
Search
Sign in
Share
File
Home
Insert
Modeling
View
Optimize
Help
Paste
Cut
Copy
Format painter
Clipboard
Get data
Excel workbook
OneLake data hub
SQL Server
Enter data
Dataverse
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Publish
Share
Average of Temperature by Date
Date
Average of Temperature
Weather Time Series Analysis and Temperature Prediction
Max of Temperature
14.82
Average of pressure
990.57
Date
3
Month
5
Year
2013
pressure and temperature
temperature
pressure
Avg. of Temperature, Avg. of Water vapor concentration and Avg. of relative humidity
Avg. of Temperature
Avg. of Water vapor concentration
Avg. of relative humidity
Filters
Visualizations
Data
Search
max_planck_weather_ts
Date
H2OC (mmol/...
max. wv (m/s)
Month
p (mbar)
rh (%)
rho (g/m**3)
sh (g/kg)
T (degC)
Tdew (degC)
Time
Tpot (K)
VPact (mbar)
VPdef (mbar)
VPmax (mbar)
Page 1
Page 1 of 1
60%
Update available (click to download)