Independent Project (Winter 2023)
Hotel Revenue Forecasting Using Machine Learning
Code Documentation

Mounting Google Drive

- The code starts by mounting the Google Drive to access files from Google Colab.

Data Preparation

- The code sets the path variable to the directory path where the Excel files are located.
- It creates a data frame named df with predefined columns to store the data extracted from the Excel files.

Extracting Data from Excel Files

- The code iterates through each file in the specified directory.
- For each Excel file, it loads the workbook, accesses the active sheet, and retrieves the actual date and business date from specific cells.
- It then iterates through each row of the sheet, starting from row 8.
- If the first column of the row contains a valid date, it creates a dictionary (row_dict) to store the data.
- The data from columns C to Z of the current row is added to row_dict.
- The row data is then appended to the df DataFrame.

Filtering the DataFrame

- The code filters the df DataFrame to keep only the rows where the business date is on or before May 5, 2023.

Reading Weather Data

- The code sets the path variable to the file location of the weather data CSV file.
- It reads the CSV file and selects specific columns (datetime, temp, humidity, precip) to create the weather_data DataFrame.

Data Transformation

- The datetime column in the weather_data DataFrame is converted to datetime format.
- Column names in the weather_data DataFrame are renamed for clarity.

Merging DataFrames

- The df DataFrame and the weather_data DataFrame are merged based on the 'Business Date' column in df and the 'datetime' column in weather_data.
- The merged DataFrame is assigned to the merged_df variable.
- The 'datetime' column is dropped from the merged DataFrame.

Final Output

- The updated merged DataFrame (merged_df) is printed, which contains the combined data from the Excel files and weather data.

Machine Learning Models Code Explanation

The code provided demonstrates two different regression models (XGBoost and Random Forest) for predicting the number of rooms sold based on various features, including temperature, humidity, and precipitation. It performs preprocessing, model training, evaluation, and visualization of the results.

XGBoost Model

1. **Preprocessing**

- The features (X) are selected from the merged_df DataFrame, including 'Actual Date', 'Business Date', 'Temperature', 'Humidity', and 'Precipitation'.
- The target variable (y) is set to 'Rooms Sold'.

2. **Convert datetime columns to datetime type**

- The 'Actual Date' and 'Business Date' columns in X are converted to datetime type using pd.to_datetime().

3. **Extract additional features from datetime columns**

- Additional features are extracted from the 'Actual Date' and 'Business Date' columns, such as year, month, and day, using the .dt accessor of the datetime columns.
- The extracted features are added as new columns to X.

4. **Drop the original datetime columns**

- The 'Actual Date' and 'Business Date' columns are dropped from X using the .drop() method.

5. **Splitting data**

- The data is split into training and test sets using train_test_split() with a test size of 0.2 and a random state of 42.

6. **XGBoost Model**

- An XGBoost regressor model is created using xgb.XGBRegressor().

7. **Cross-validation**

- Cross-validation is performed on the training set using cross_val_score() with 5-fold cross-validation and the negative mean squared error (MSE) as the scoring metric.
- The negative MSE scores are converted to positive.

8. Train the model

- The model is trained on the entire training set using the .fit() method.

9. Predictions on the validation and test sets

- Predictions are made on the validation set using the .predict() method and stored in y_val_pred.
- Predictions are made on the test set and stored in y_test_pred.

10. Evaluation and Performance Metrics

- Mean squared error (MSE), root mean squared error (RMSE), and R-squared (R2) scores are calculated for the validation and test sets using appropriate evaluation metrics (mean_squared_error(), mean_absolute_error(), r2_score()).
- The results are printed.

11. Visualization

- Scatter plot: The predicted values vs. the actual values for the validation set are plotted, with a diagonal line representing perfect predictions.
- Residual plot: The residuals (difference between actual and predicted values) are plotted against the actual values for the validation set.
- Feature importances: The importance of each feature in the XGBoost model is plotted as a bar chart.

Random Forest Regressor Model

1. **Preprocessing**

- The features (X) and target variable (y) are prepared in the same way as in the XGBoost model.

2. **Convert datetime columns to datetime type**

- The 'Actual Date' and 'Business Date' columns in X are converted to datetime type using pd.to_datetime().

3. **Extract additional features from datetime columns**

- Additional features are extracted from the 'Actual Date' and 'Business Date' columns, such as year, month, and day, using the .dt accessor of the datetime columns.
- The extracted features are added as new columns to X.

4. **Drop the original datetime columns**

- The 'Actual Date' and 'Business Date' columns are dropped from X using the .drop() method.

5. **Splitting data**

- The data is split into training and test sets using train_test_split() with a test size of 0.2 and a random state of 42.

6. **Random Forest Regressor Model**

- A Random Forest regressor model is created using RandomForestRegressor().

7. **Cross-validation**

- Cross-validation is performed on the training set using cross_val_score() with 5-fold cross-validation and the negative mean squared error (MSE) as the scoring metric.
- The negative MSE scores are converted to positive.

8. Train the model

- The model is trained on the entire training set using the .fit() method.

9. Predictions on the validation and test sets

- Predictions are made on the validation set using the .predict() method and stored in y_val_pred.
- Predictions are made on the test set and stored in y_test_pred.

10. Evaluation and Performance Metrics

- Mean squared error (MSE), root mean squared error (RMSE), and R-squared (R2) scores are calculated for the validation and test sets using appropriate evaluation metrics (mean_squared_error(), mean_absolute_error(), r2_score()).
- The results are printed.

11. Visualization

- Bar chart: Performance metrics (MSE, RMSE, R2 score) for the validation and test sets are plotted side by side.
- Feature importances: The importance of each feature in the Random Forest model is plotted as a horizontal bar chart.

Conclusion

The provided code demonstrates how to train and evaluate regression models (XGBoost and Random Forest) to predict the number of rooms sold based on various features. It includes preprocessing steps, model training, evaluation using performance metrics, and

visualization of results. These steps can be helpful for analyzing and predicting room sales in the hospitality industry or similar scenarios.