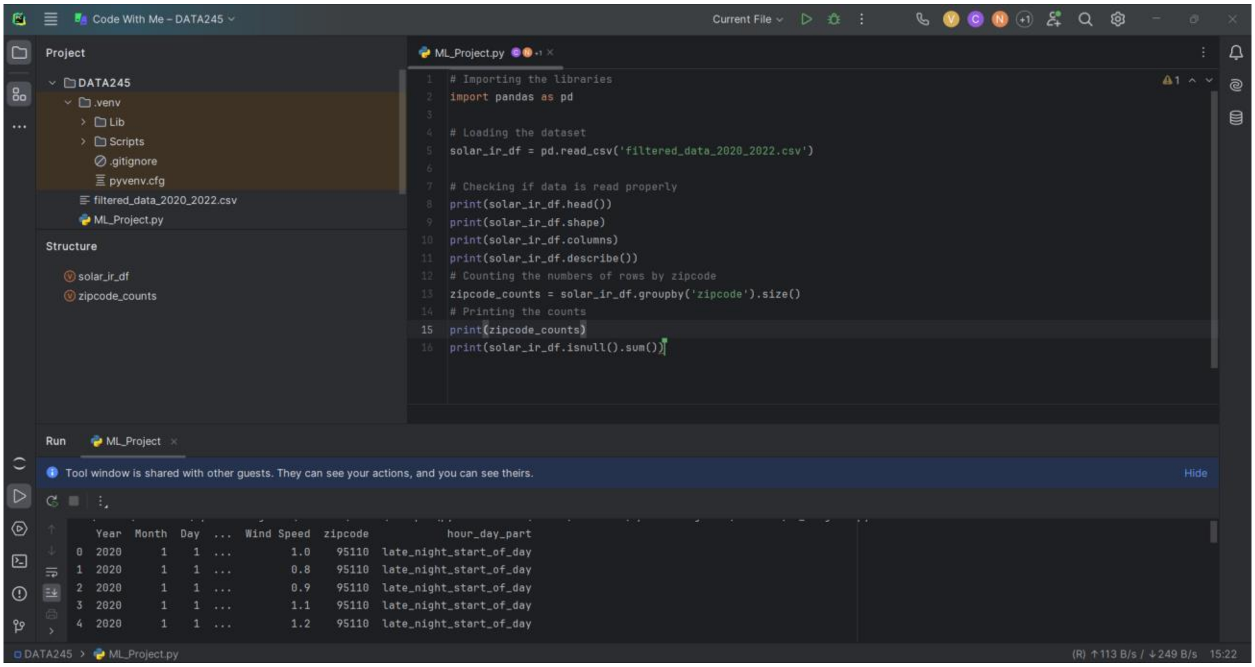
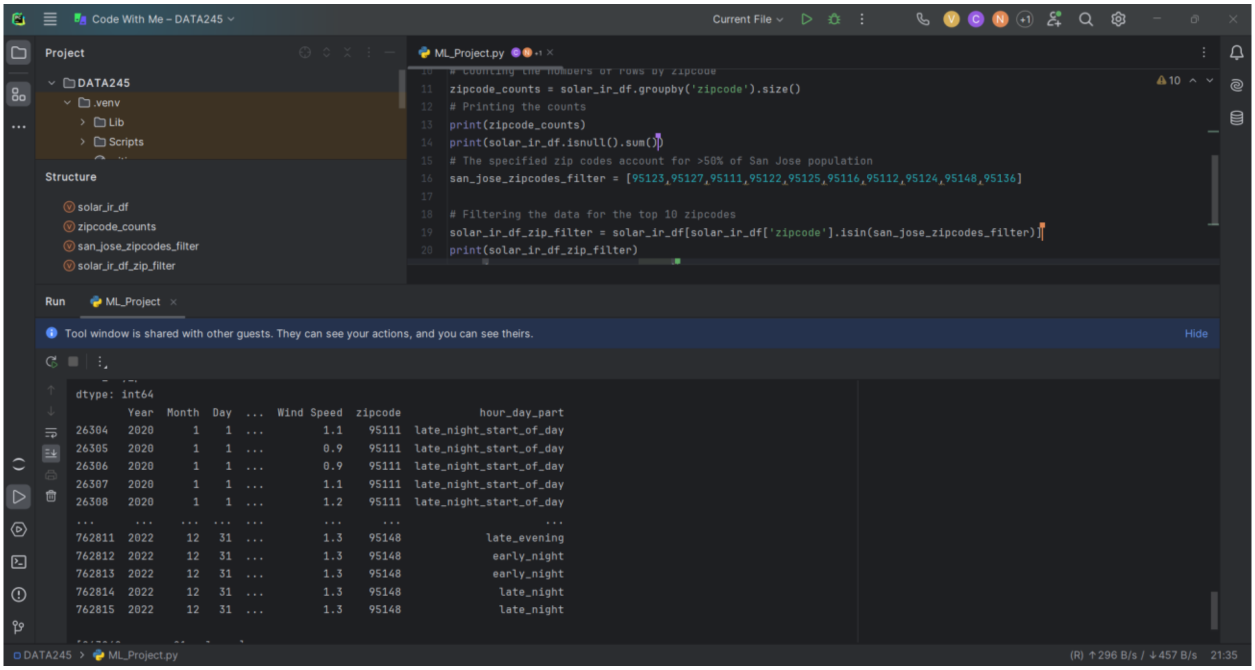
GitHub copilot practiced by Group 3.
Project Name is DATA 245
We created the environment and added data in csv and during a call we simultaneously did peer review of the code to ensure our code quality is consistent.



```python
# Importing the libraries
import pandas as pd

# Loading the dataset
solar_ir_df = pd.read_csv('filtered_data_2020_2022.csv')

# Checking if data is read properly
print(solar_ir_df.head())
print(solar_ir_df.shape)
print(solar_ir_df.columns)
print(solar_ir_df.describe())
# Counting the numbers of rows by zipcode
zipcode_counts = solar_ir_df.groupby('zipcode').size()
# Printing the counts
print(zipcode_counts)
print(solar_ir_df.isnull().sum())
```

Tool window is shared with other guests. They can see your actions, and you can see theirs.

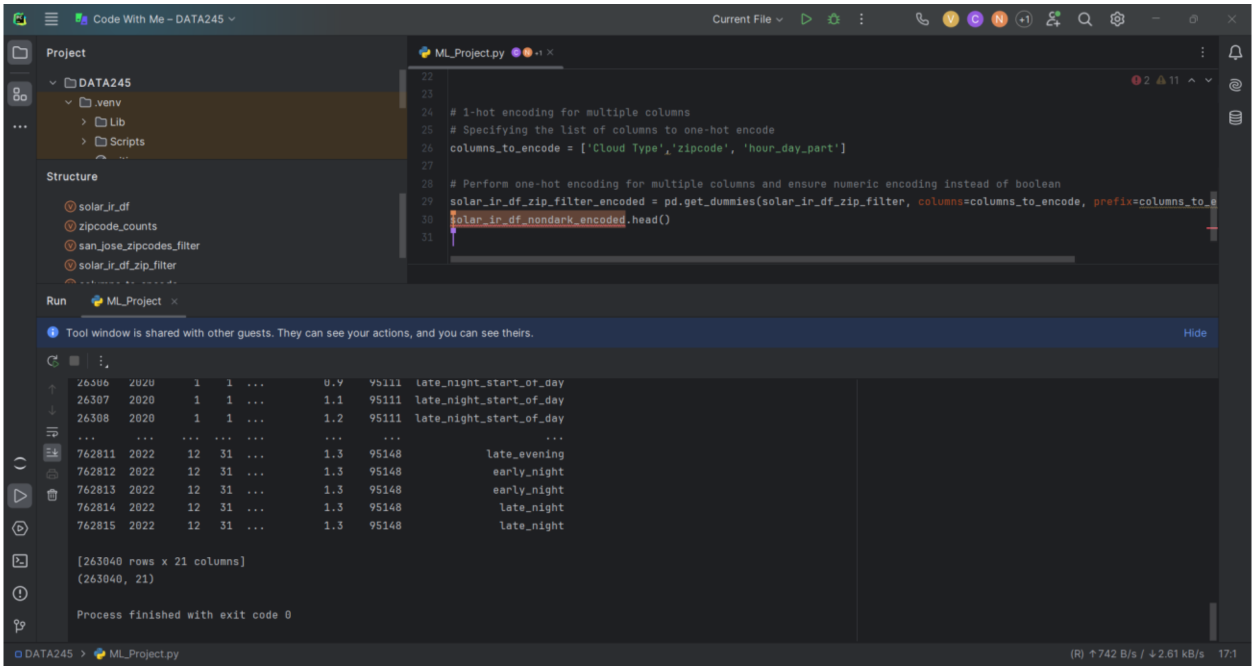
```
   Year  Month  Day  ...  Wind Speed  zipcode            hour_day_part
0  2020      1    1  ...         1.0    95110  late_night_start_of_day
1  2020      1    1  ...         0.8    95110  late_night_start_of_day
2  2020      1    1  ...         0.9    95110  late_night_start_of_day
3  2020      1    1  ...         1.1    95110  late_night_start_of_day
4  2020      1    1  ...         1.2    95110  late_night_start_of_day
```



```python
# Counting the numbers of rows by zipcode
zipcode_counts = solar_ir_df.groupby('zipcode').size()
# Printing the counts
print(zipcode_counts)
print(solar_ir_df.isnull().sum())
# The specified zip codes account for >50% of San Jose population
san_jose_zipcodes_filter = [95123,95127,95111,95122,95125,95116,95112,95124,95148,95136]

# Filtering the data for the top 10 zipcodes
solar_ir_df_zip_filter = solar_ir_df[solar_ir_df['zipcode'].isin(san_jose_zipcodes_filter)]
print(solar_ir_df_zip_filter)
```

Tool window is shared with other guests. They can see your actions, and you can see theirs.

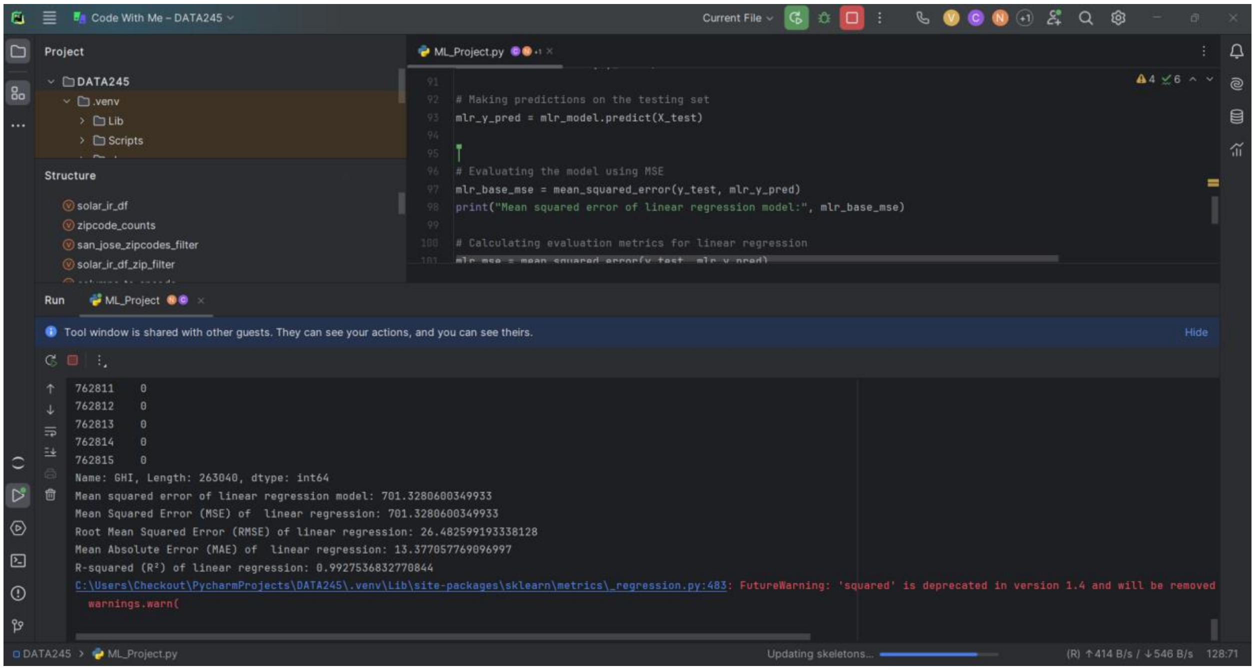
```
dtype: int64
        Year  Month  Day  ...  Wind Speed  zipcode            hour_day_part
26304   2020      1    1  ...         1.1    95111  late_night_start_of_day
26305   2020      1    1  ...         0.9    95111  late_night_start_of_day
26306   2020      1    1  ...         0.9    95111  late_night_start_of_day
26307   2020      1    1  ...         1.1    95111  late_night_start_of_day
26308   2020      1    1  ...         1.2    95111  late_night_start_of_day
...      ...    ...  ...  ...         ...      ...                      ...
762811  2022     12   31  ...         1.3    95148             late_evening
762812  2022     12   31  ...         1.3    95148              early_night
762813  2022     12   31  ...         1.3    95148              early_night
762814  2022     12   31  ...         1.3    95148               late_night
762815  2022     12   31  ...         1.3    95148               late_night
```



```python
# 1-hot encoding for multiple columns
# Specifying the list of columns to one-hot encode
columns_to_encode = ['Cloud Type','zipcode', 'hour_day_part']

# Perform one-hot encoding for multiple columns and ensure numeric encoding instead of boolean
solar_ir_df_zip_filter_encoded = pd.get_dummies(solar_ir_df_zip_filter, columns=columns_to_encode, prefix=columns_to_e
solar_ir_df_nondark_encoded.head()
```

Tool window is shared with other guests. They can see your actions, and you can see theirs.

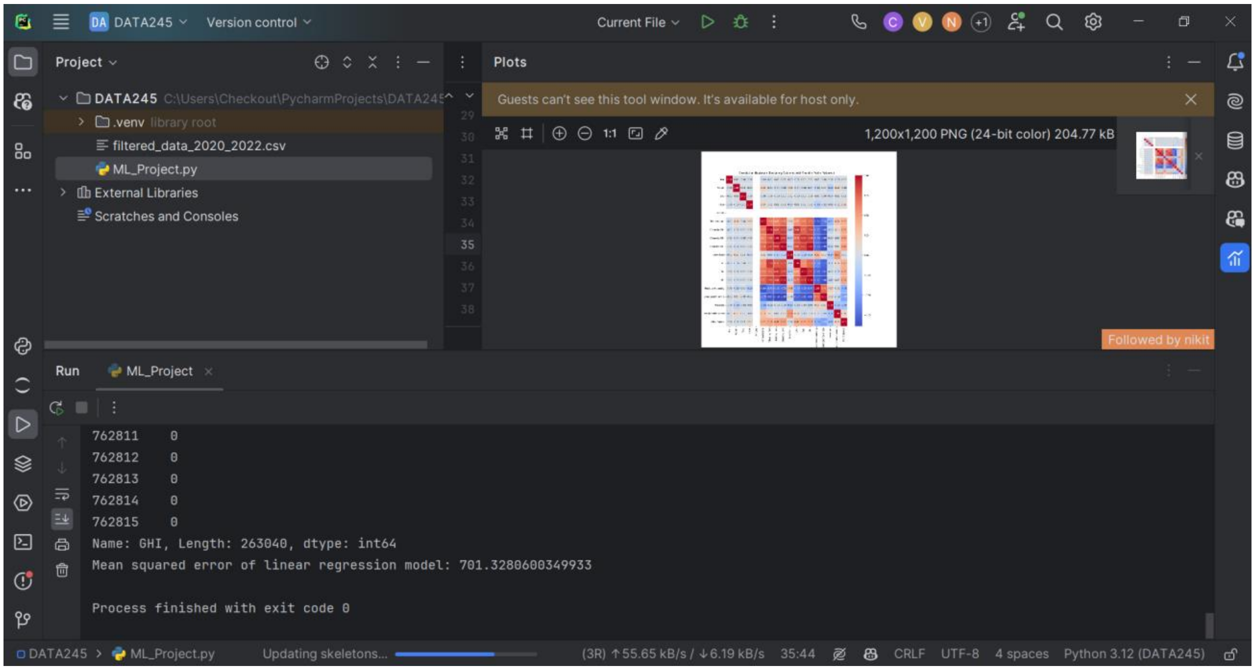
```
26306   2020     1    1  ...      0.9   95111  late_night_start_of_day
26307   2020     1    1  ...      1.1   95111  late_night_start_of_day
26308   2020     1    1  ...      1.2   95111  late_night_start_of_day
...      ...   ...  ...  ...      ...     ...                      ...
762811  2022    12   31  ...      1.3   95148             late_evening
762812  2022    12   31  ...      1.3   95148              early_night
762813  2022    12   31  ...      1.3   95148              early_night
762814  2022    12   31  ...      1.3   95148               late_night
762815  2022    12   31  ...      1.3   95148               late_night

[263040 rows x 21 columns]
(263040, 21)

Process finished with exit code 0
```



```python

# Making predictions on the testing set
mlr_y_pred = mlr_model.predict(X_test)



# Evaluating the model using MSE
mlr_base_mse = mean_squared_error(y_test, mlr_y_pred)
print("Mean squared error of linear regression model:", mlr_base_mse)

# Calculating evaluation metrics for linear regression
mlr_mse = mean_squared_error(y_test, mlr_y_pred)
```

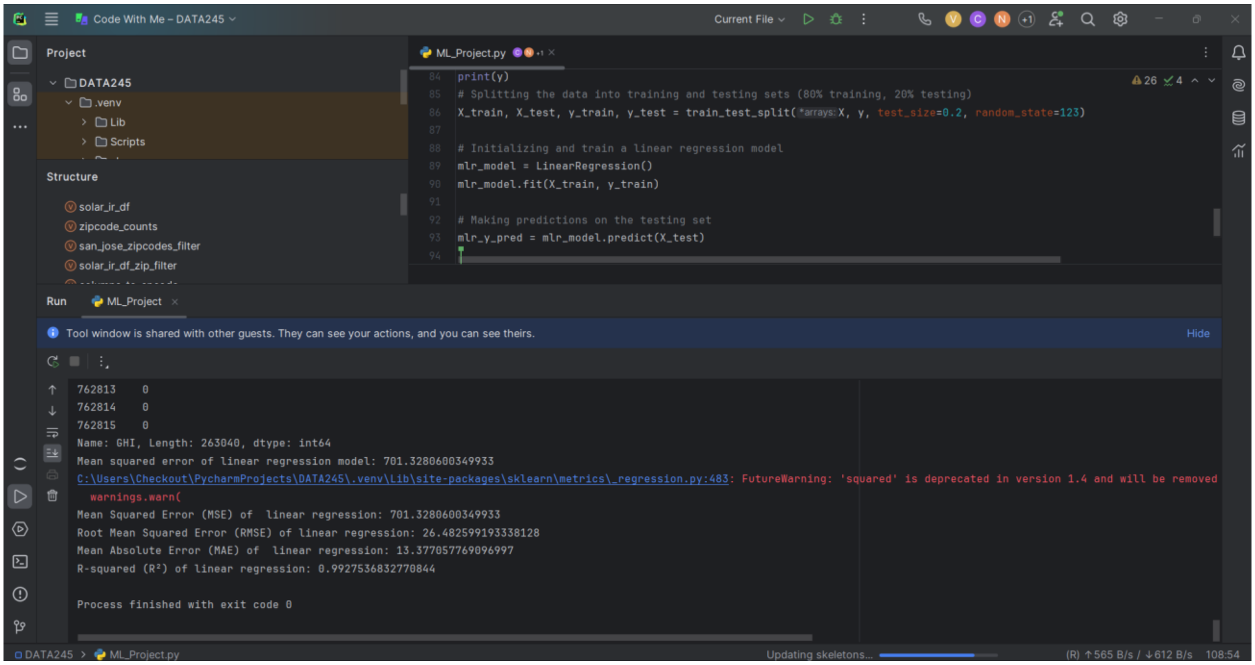
Tool window is shared with other guests. They can see your actions, and you can see theirs.

```
762811    0
762812    0
762813    0
762814    0
762815    0
Name: GHI, Length: 263040, dtype: int64
Mean squared error of linear regression model: 701.3280600349933
Mean Squared Error (MSE) of  linear regression: 701.3280600349933
Root Mean Squared Error (RMSE) of linear regression: 26.482599193338128
Mean Absolute Error (MAE) of  linear regression: 13.377057769096997
R-squared (R²) of linear regression: 0.9927536832770844
C:\Users\Checkout\PycharmProjects\DATA245\.venv\Lib\site-packages\sklearn\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed
  warnings.warn(
```



Guests can't see this tool window. It's available for host only.

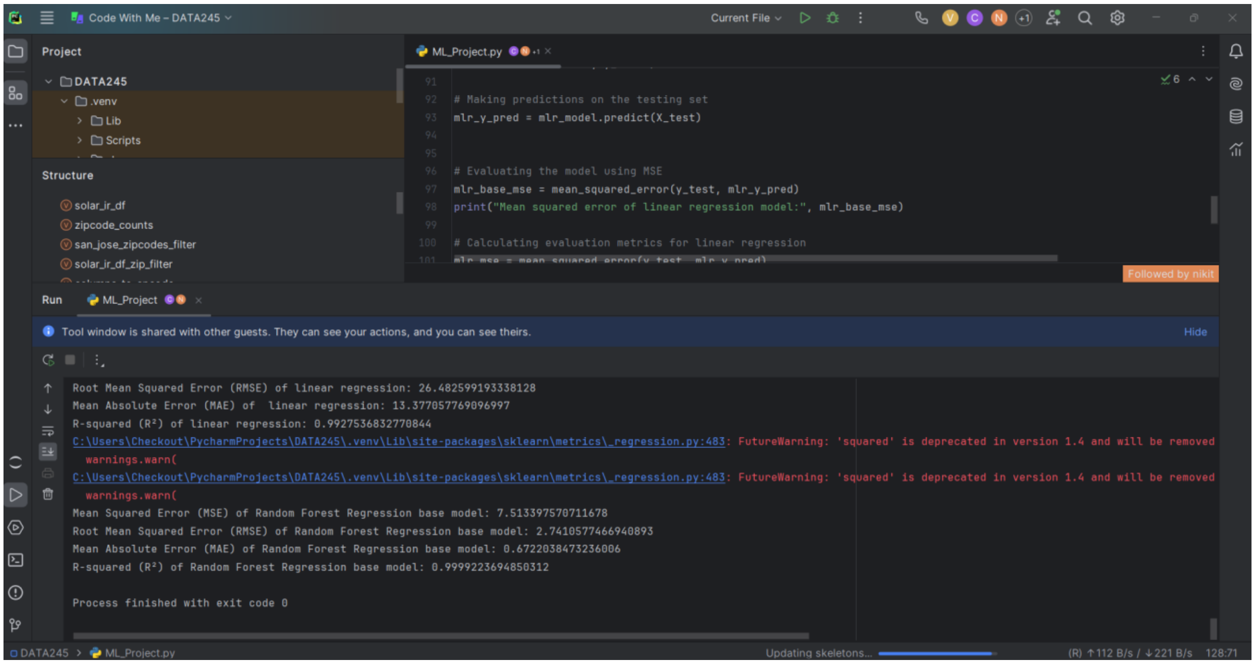
```
762811    0
762812    0
762813    0
762814    0
762815    0
Name: GHI, Length: 263040, dtype: int64
Mean squared error of linear regression model: 701.3280600349933

Process finished with exit code 0
```



```python
print(y)
# Splitting the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.2, random_state=123)

# Initializing and train a linear regression model
mlr_model = LinearRegression()
mlr_model.fit(X_train, y_train)

# Making predictions on the testing set
mlr_y_pred = mlr_model.predict(X_test)
```

Tool window is shared with other guests. They can see your actions, and you can see theirs.

```
762813    0
762814    0
762815    0
Name: GHI, Length: 263040, dtype: int64
Mean squared error of linear regression model: 701.3280600349933
C:\Users\Checkout\PycharmProjects\DATA245\.venv\Lib\site-packages\sklearn\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed
  warnings.warn(
Mean Squared Error (MSE) of  linear regression: 701.3280600349933
Root Mean Squared Error (RMSE) of linear regression: 26.482599193338128
Mean Absolute Error (MAE) of  linear regression: 13.377057769096997
R-squared (R²) of linear regression: 0.9927536832770844

Process finished with exit code 0
```



```python

# Making predictions on the testing set
mlr_y_pred = mlr_model.predict(X_test)


# Evaluating the model using MSE
mlr_base_mse = mean_squared_error(y_test, mlr_y_pred)
print("Mean squared error of linear regression model:", mlr_base_mse)

# Calculating evaluation metrics for linear regression
mlr_mse = mean_squared_error(y_test, mlr_y_pred)
```

Tool window is shared with other guests. They can see your actions, and you can see theirs.

```
Root Mean Squared Error (RMSE) of linear regression: 26.482599193338128
Mean Absolute Error (MAE) of  linear regression: 13.377057769096997
R-squared (R²) of linear regression: 0.9927536832770844
C:\Users\Checkout\PycharmProjects\DATA245\.venv\Lib\site-packages\sklearn\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed
  warnings.warn(
C:\Users\Checkout\PycharmProjects\DATA245\.venv\Lib\site-packages\sklearn\metrics\_regression.py:483: FutureWarning: 'squared' is deprecated in version 1.4 and will be removed
  warnings.warn(
Mean Squared Error (MSE) of Random Forest Regression base model: 7.513397570711678
Root Mean Squared Error (RMSE) of Random Forest Regression base model: 2.7410577466940893
Mean Absolute Error (MAE) of Random Forest Regression base model: 0.6722038473236006
R-squared (R²) of Random Forest Regression base model: 0.9999223694850312

Process finished with exit code 0
```