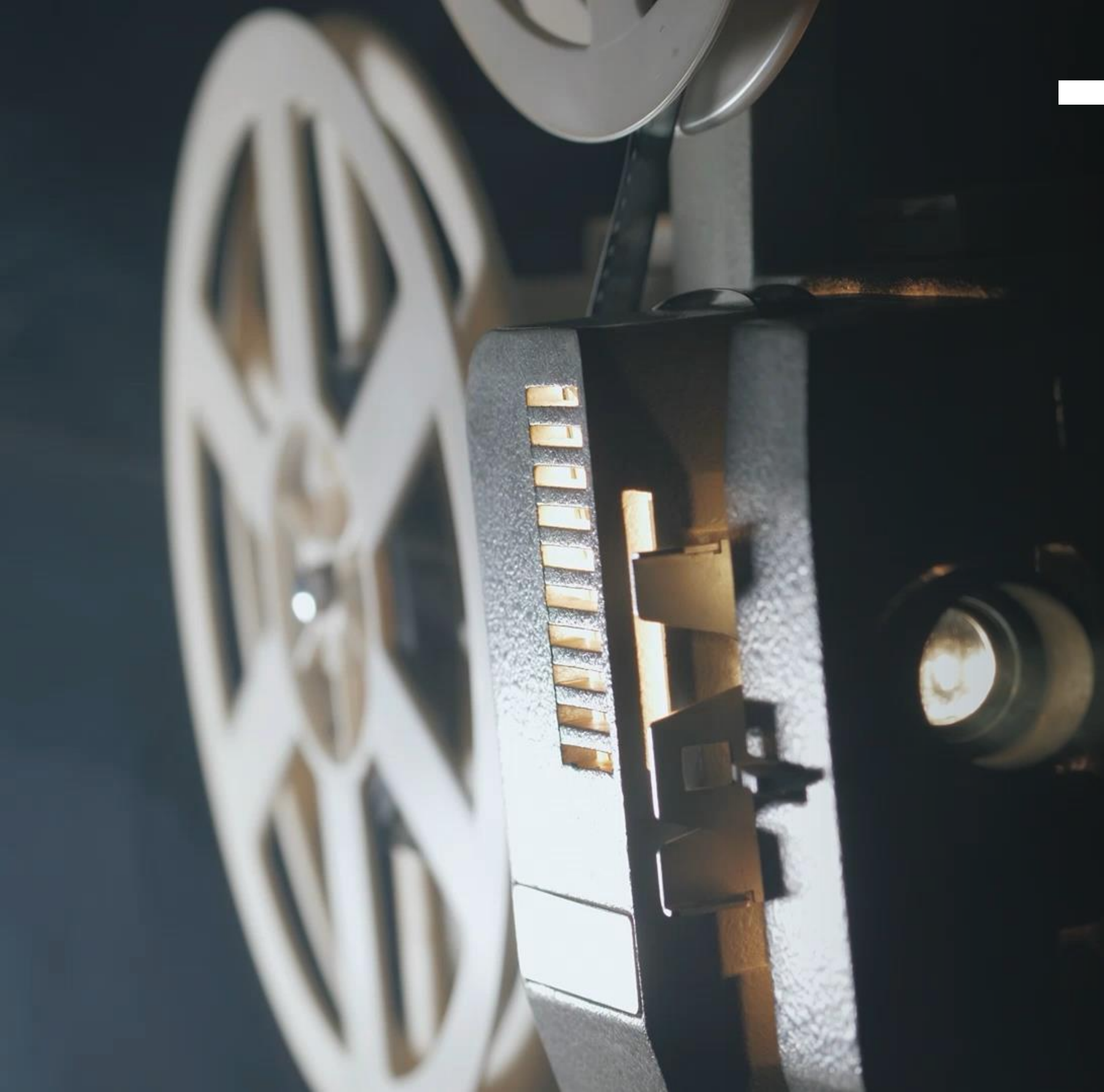
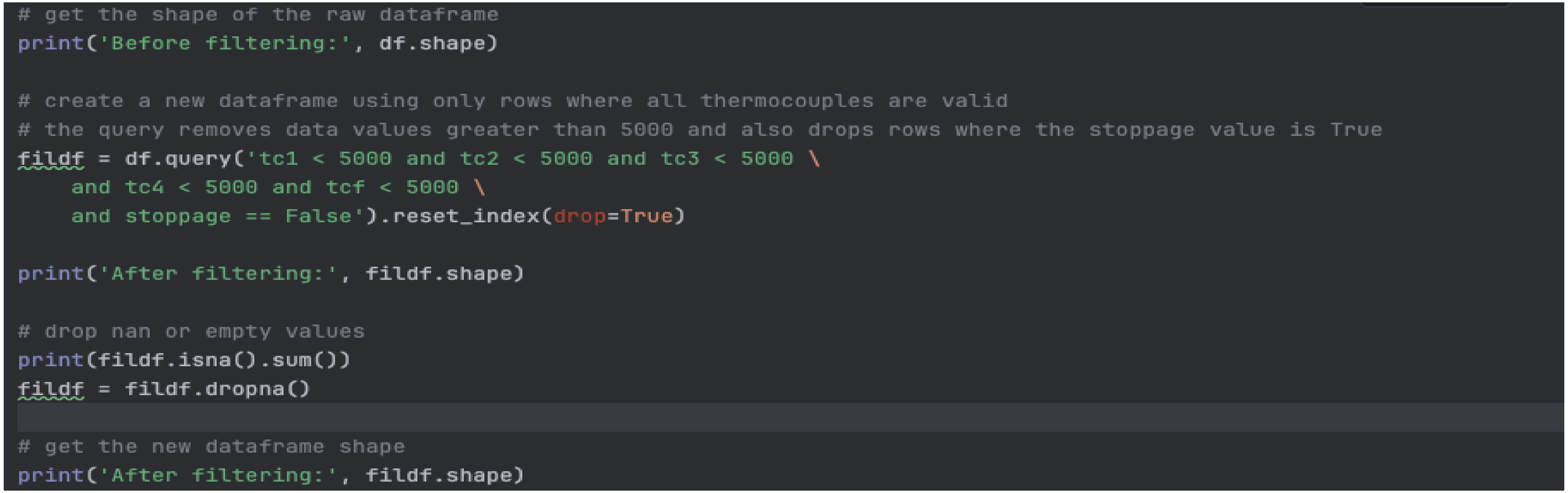


Machine Learning for Industrial Combustion System Optimization

Predicting NOx Emissions, CO2, and Temperature

Task 1: Data Cleaning (98.3% retention)

- Issues in data
  - Failed Thermocouples - Sensors showing impossible readings like 9999 C
  - Downtime periods - Times when the system was shutdown
  - Missing Values - Any NaN Values or Empty Values

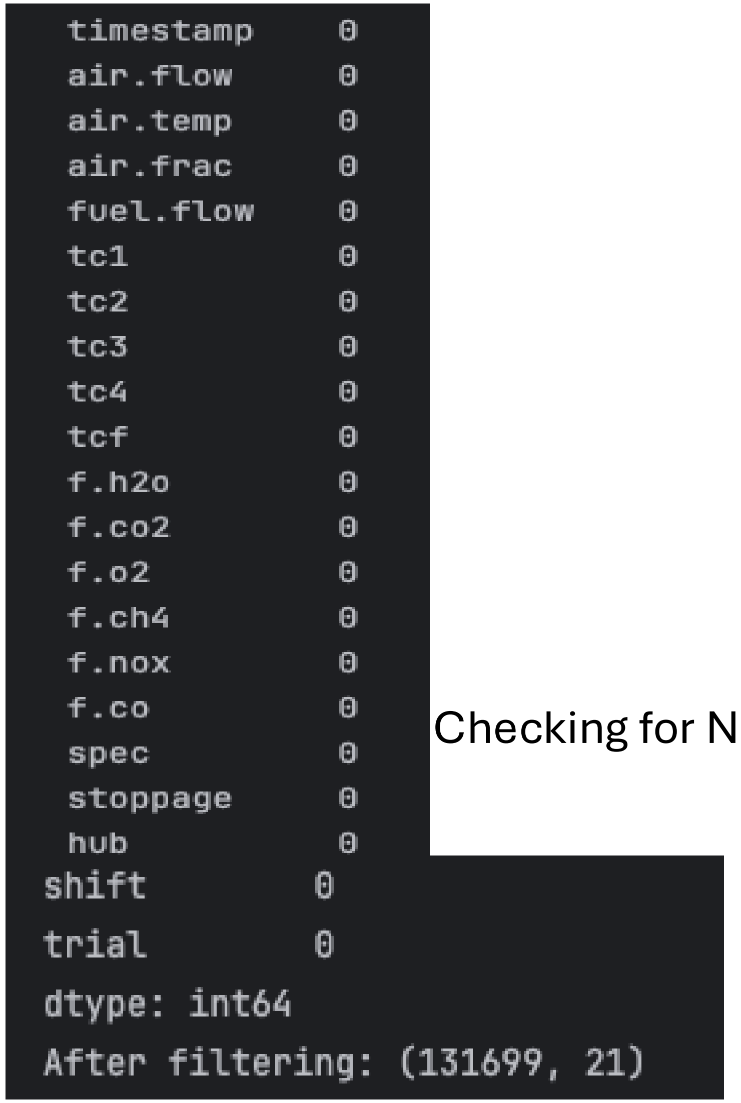


```
# get the shape of the raw dataframe
print('Before filtering:', df.shape)

# create a new dataframe using only rows where all thermocouples are valid
# the query removes data values greater than 5000 and also drops rows where the stoppage value is True
fildf = df.query('tc1 < 5000 and tc2 < 5000 and tc3 < 5000 \
    and tc4 < 5000 and tcf < 5000 \
    and stoppage == False').reset_index(drop=True)

print('After filtering:', fildf.shape)

# drop nan or empty values
print(fildf.isna().sum())
fildf = fildf.dropna()

# get the new dataframe shape
print('After filtering:', fildf.shape)
```

Results

Dataset:

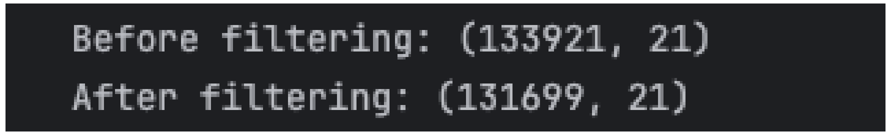
- **4 Inputs:** Air flow, Air temperature, Oxygen fraction, Fuel flow
- **3 Outputs:** NOx emissions, Flue temperature, CO2 emissions



```
  timestamp    0
  air.flow     0
  air.temp     0
  air.frac     0
  fuel.flow    0
  tc1          0
  tc2          0
  tc3          0
  tc4          0
  tcf          0
  f.h2o        0
  f.co2        0
  f.o2         0
  f.ch4        0
  f.nox        0
  f.co         0
  spec         0
  stoppage     0
  hub          0
shift        0
trial        0
dtype: int64
After filtering: (131699, 21)
```

Checking for NaN or Empty values

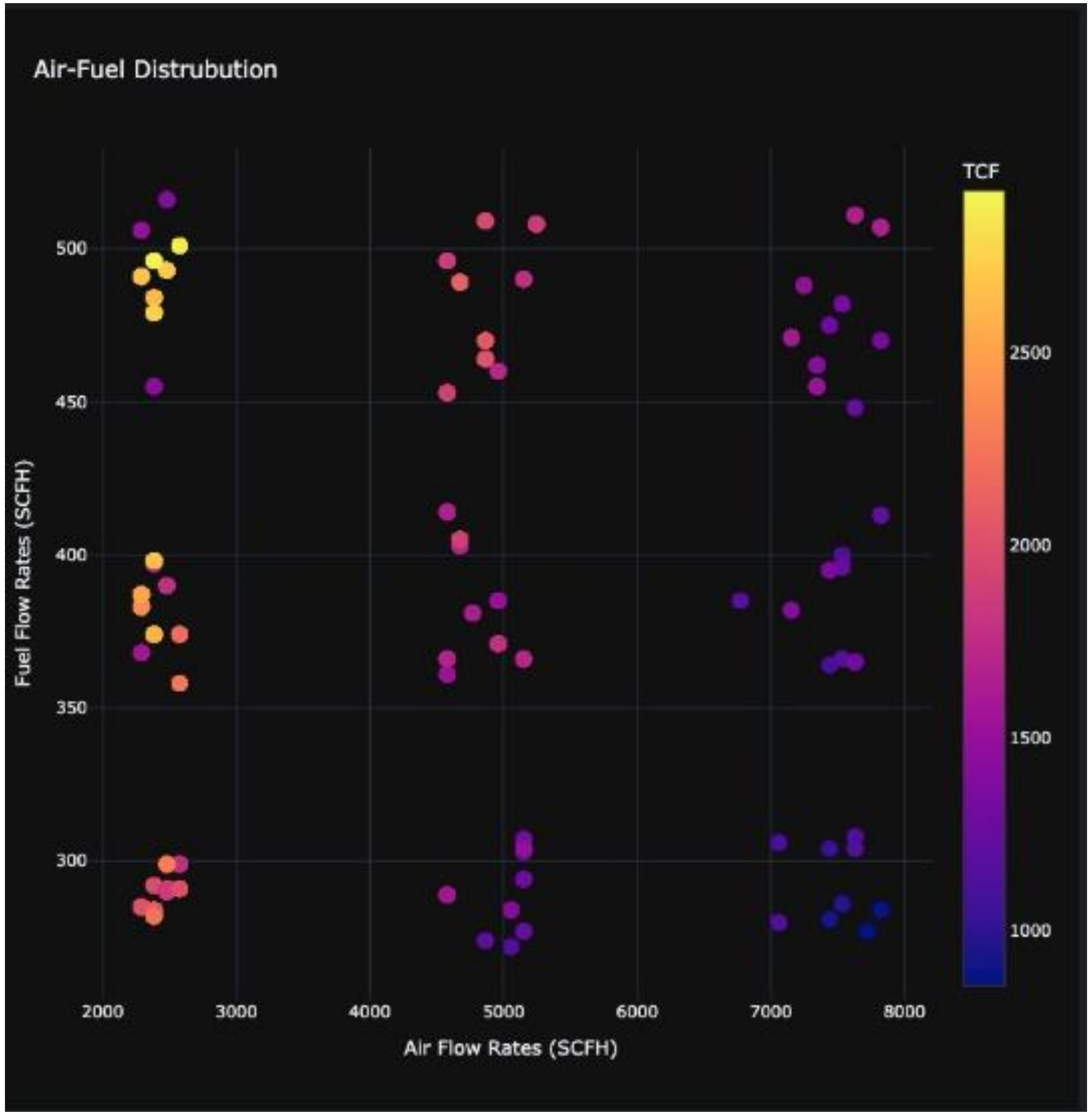
After removing failed thermocouples and system stoppage



```
Before filtering: (133921, 21)
After filtering: (131699, 21)
```

Task 2 - Exploratory Analysis Key Findings

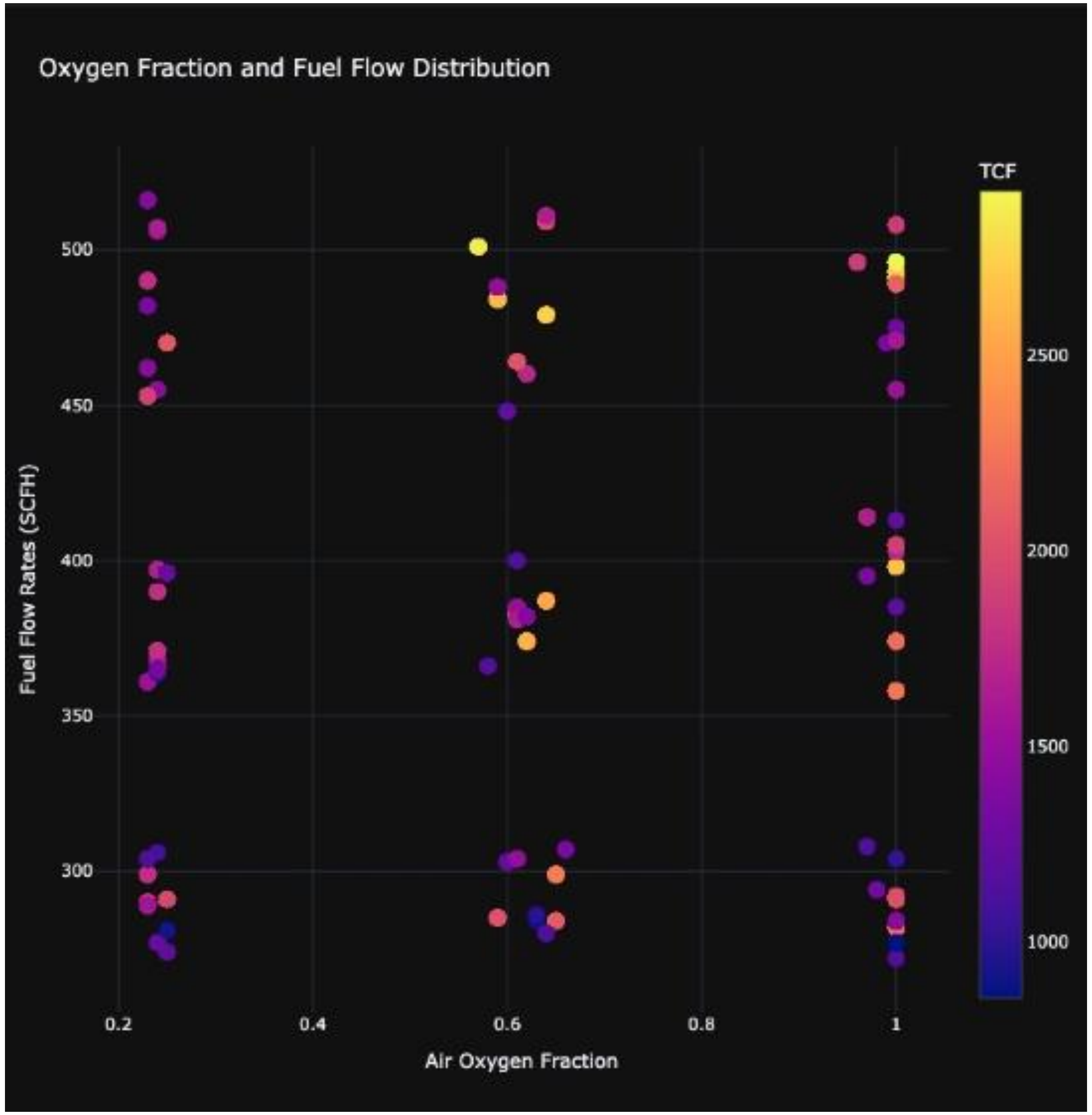
1. **Air-Fuel Ratio vs. Temperature**
   - **Low air flow** (2,500 SCFH) → High temp (2,000-2,700°C) - Yellow/Orange
   - **High air flow** (7,500 SCFH) → Low temp (1,000-1,500°C) - Blue/Purple
   - Air-fuel ratio is the dominant factor for temperature



EAKF - Continued

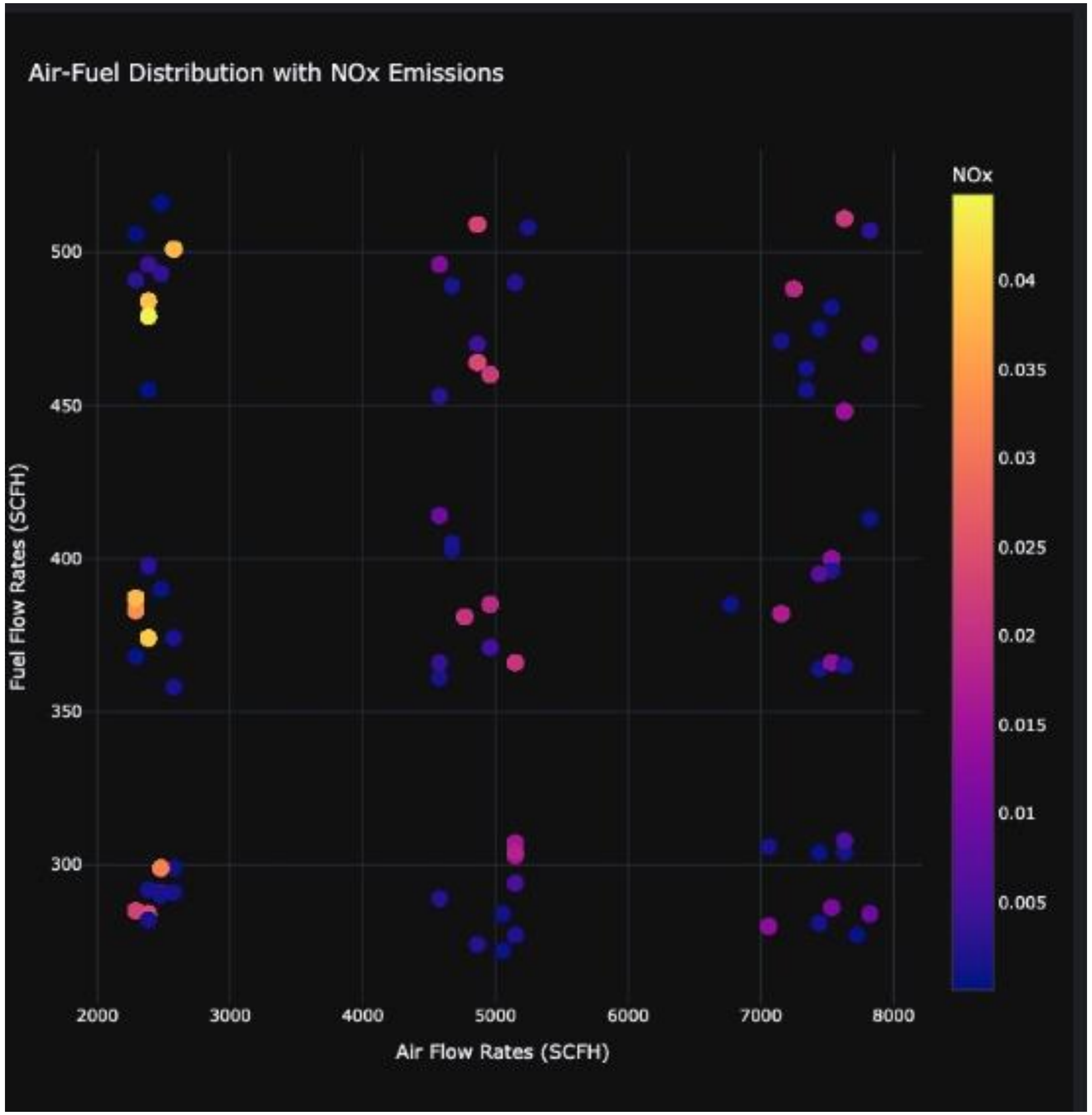
2. Oxygen Fraction vs. Temperature

- Pure oxygen (1.0) → 2,700°C (yellow)
- Normal air (0.21) → 1,200°C (blue)
- 1,000°C+ temperature increase from oxygen enrichment

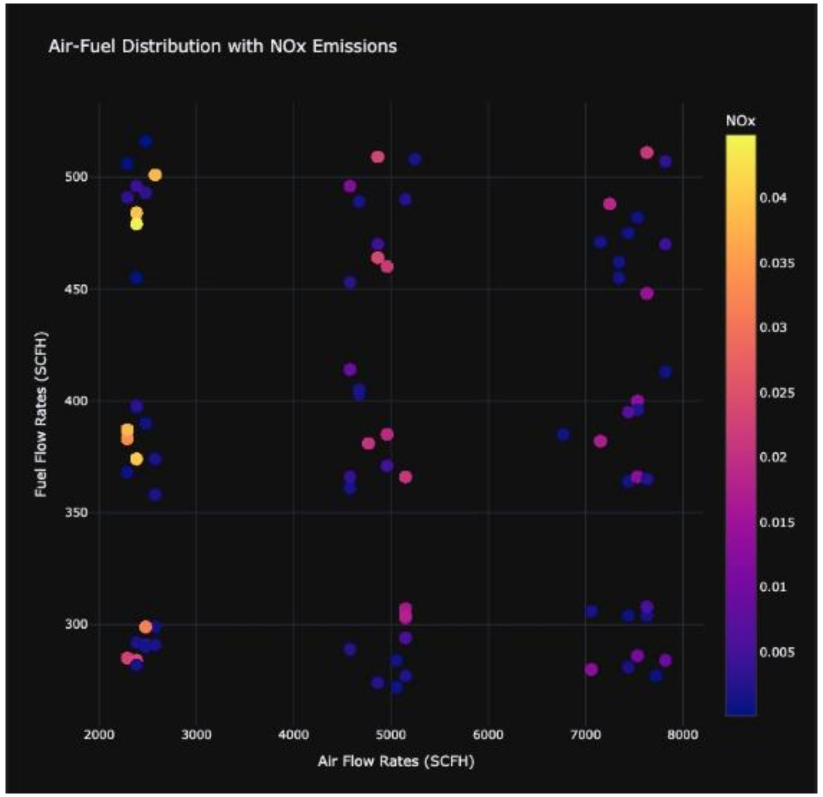


EAKF - Continued

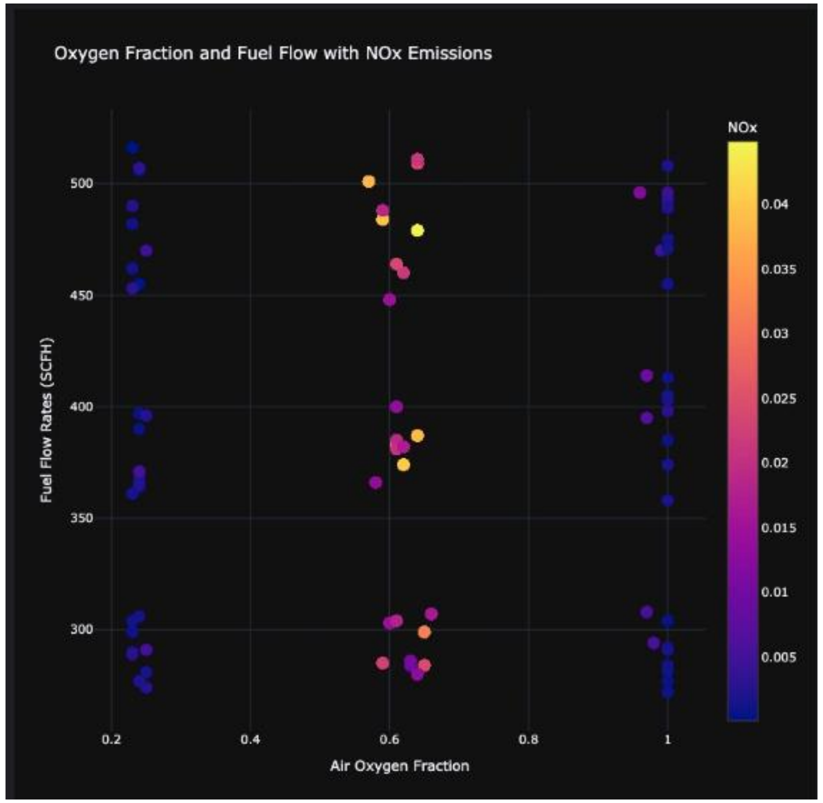
3. Air-Fuel Ratio vs. NOx Emission
   - Low air flow → High NOx (0.04-0.045) - Yellow
   - High air flow → Low NOx (0.002-0.01) - Blue
   - Direct correlation: High Temperature = High NOx



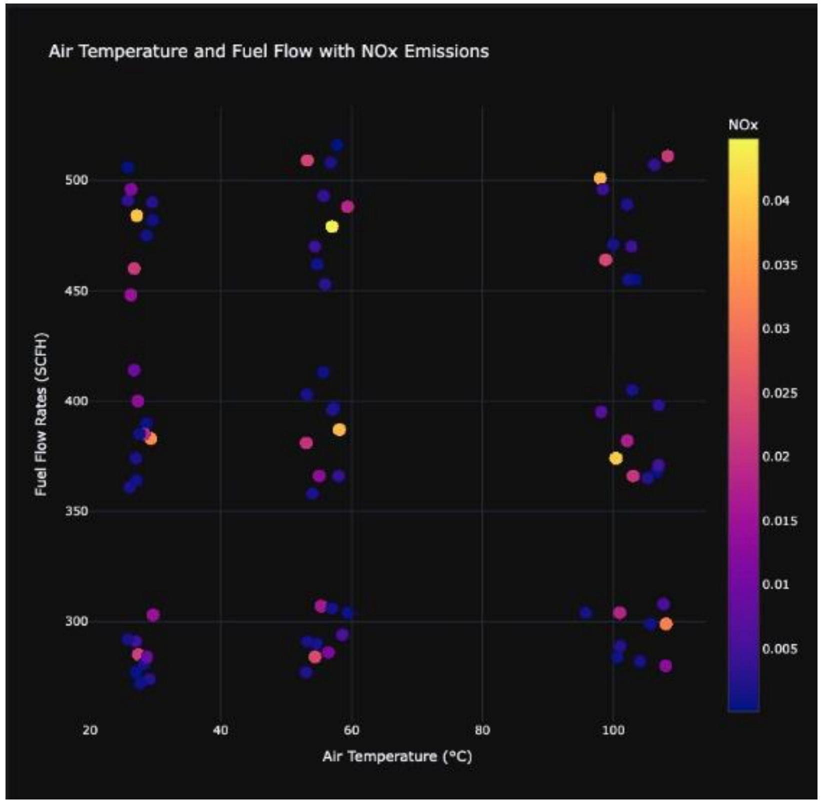
NOx Emissions Analysis (All Factors)



Air-Fuel vs NOx

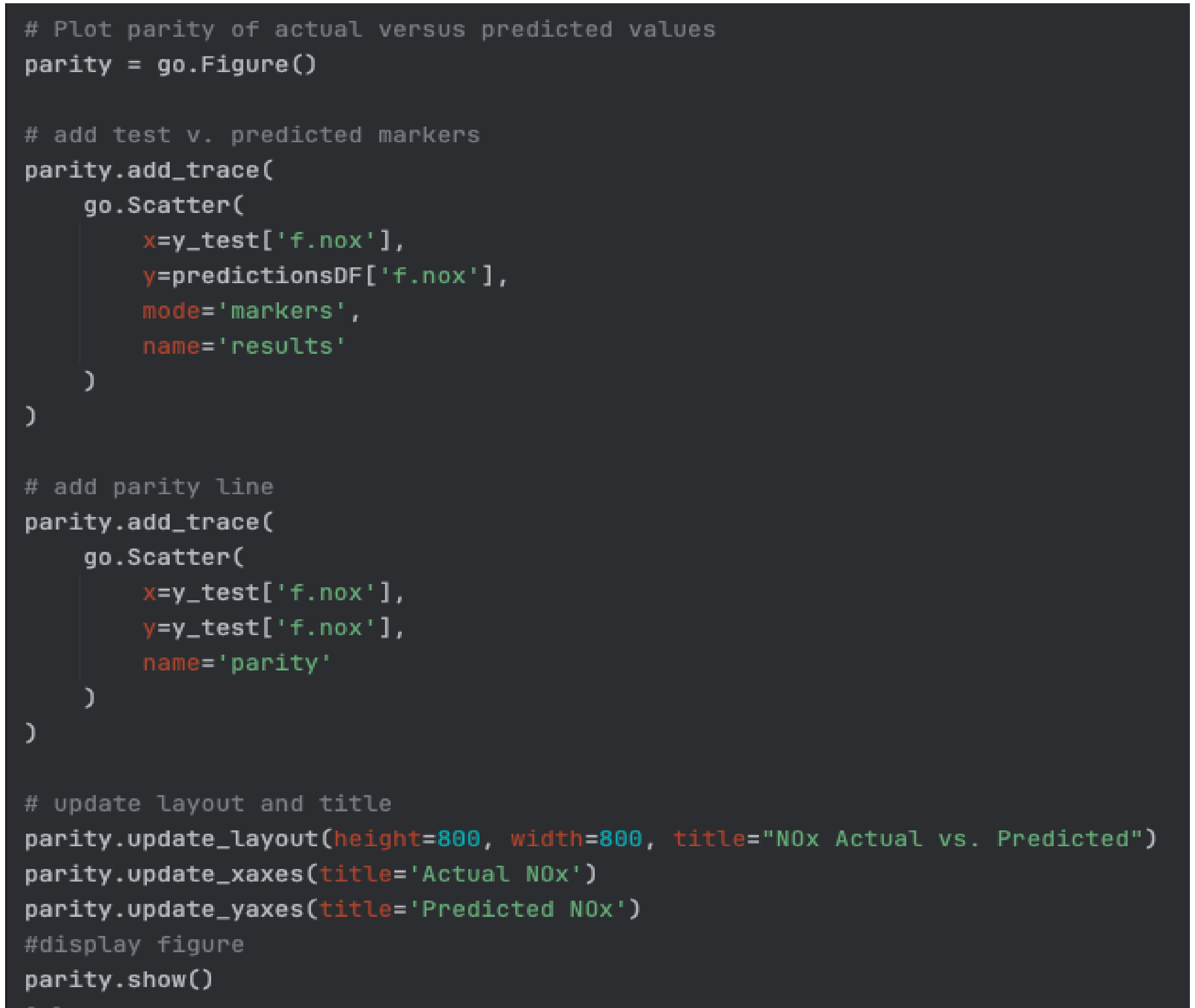


Oxygen vs NOx

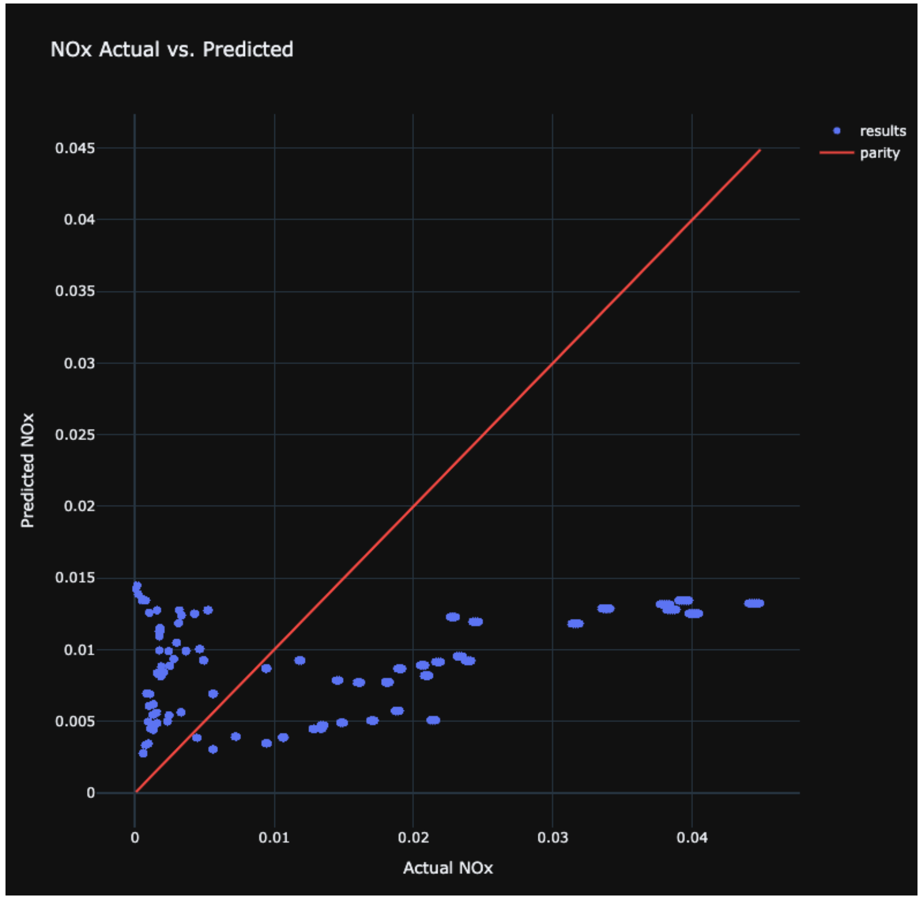


Air-Temp vs NOx

Task 3 - Linear Regression - Predict NOx

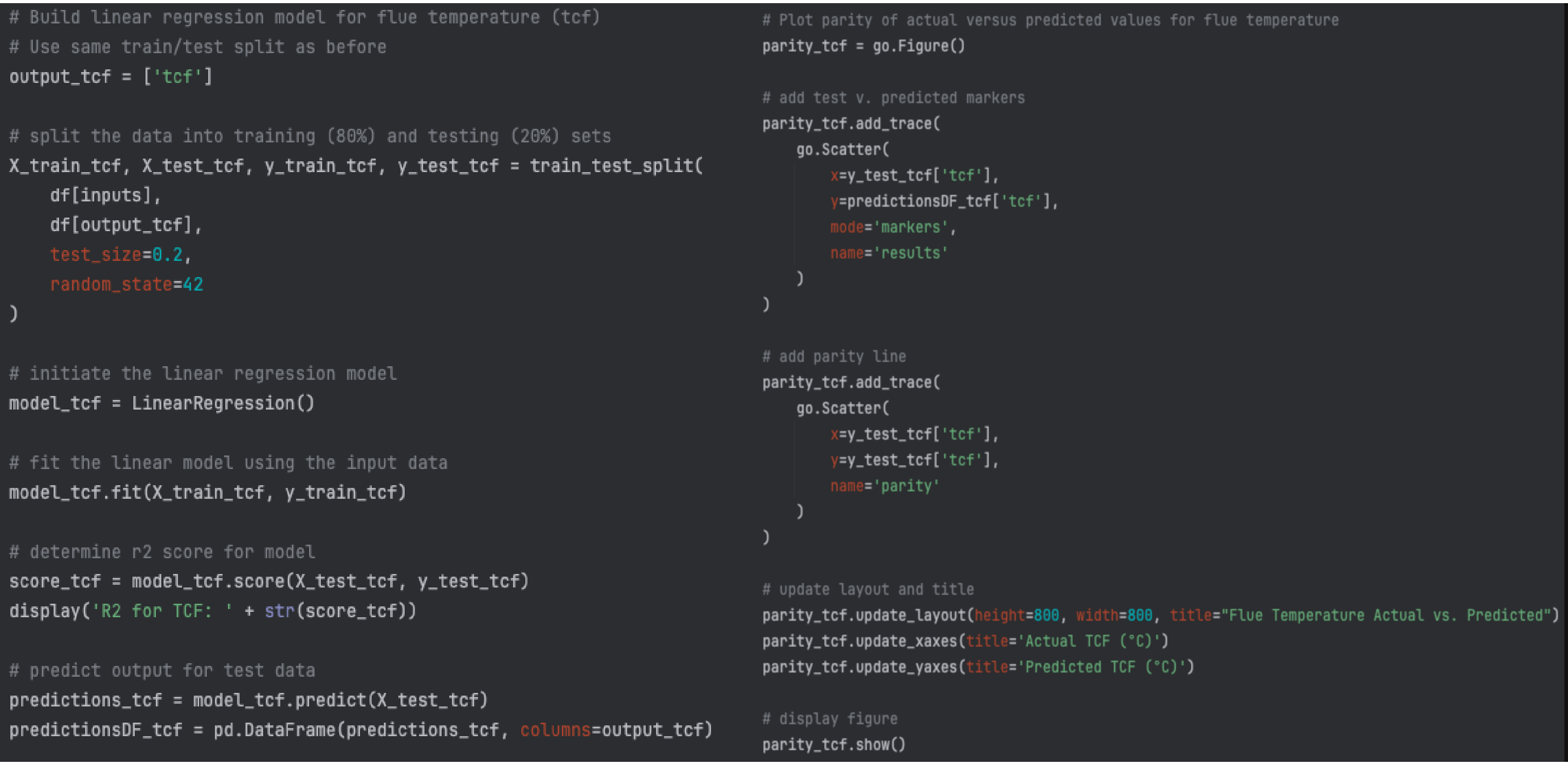


```
# Plot parity of actual versus predicted values
parity = go.Figure()

# add test v. predicted markers
parity.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=predictionsDF['f.nox'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=y_test['f.nox'],
        name='parity'
    )
)

# update layout and title
parity.update_layout(height=800, width=800, title="NOx Actual vs. Predicted")
parity.update_xaxes(title='Actual NOx')
parity.update_yaxes(title='Predicted NOx')
#display figure
parity.show()
```



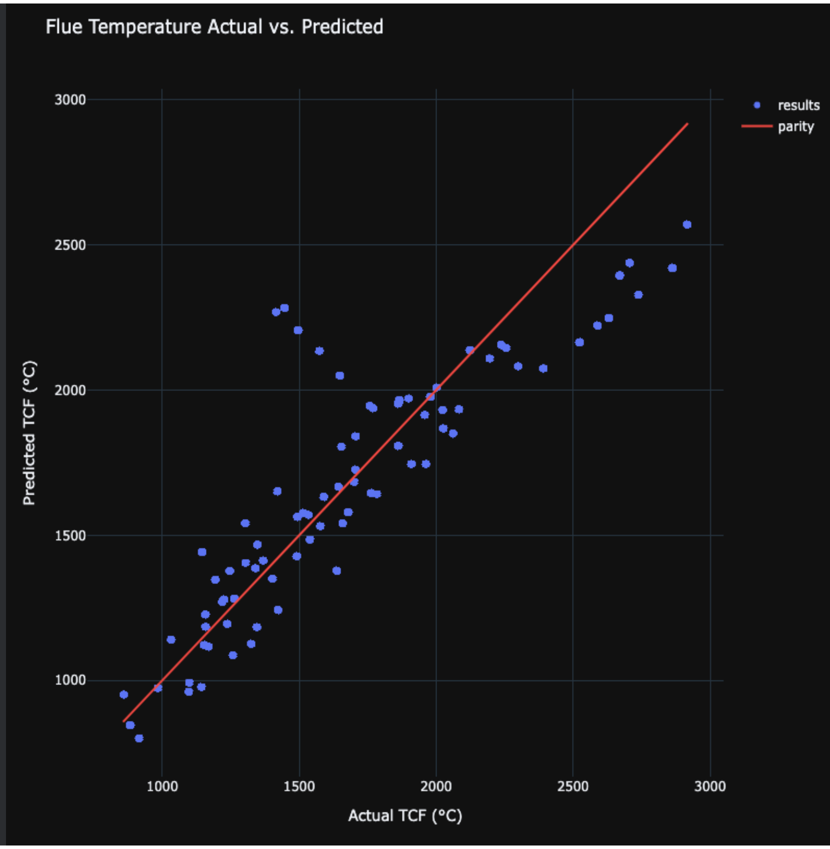
'R2: 0.099349427561162'

Predict Flue temperature



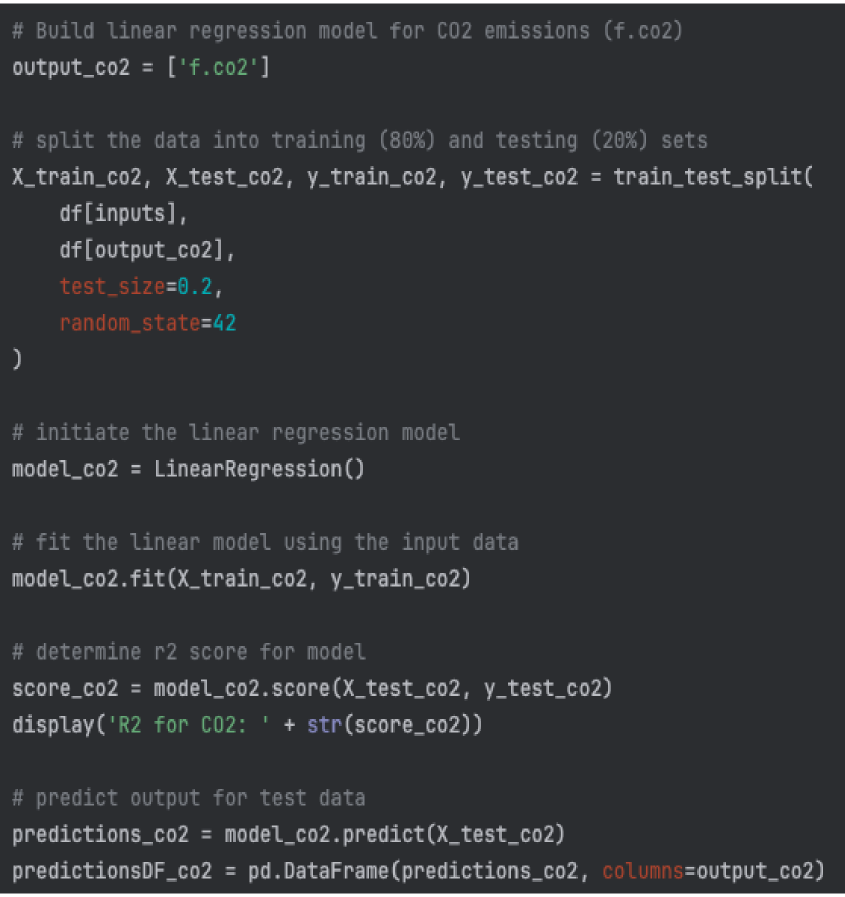
```
# Build linear regression model for flue temperature (tcf)
# Use same train/test split as before
output_tcf = ['tcf']

# split the data into training (80%) and testing (20%) sets
X_train_tcf, X_test_tcf, y_train_tcf, y_test_tcf = train_test_split(
    df[inputs],
    df[output_tcf],
    test_size=0.2,
    random_state=42
)

# initiate the linear regression model
model_tcf = LinearRegression()

# fit the linear model using the input data
model_tcf.fit(X_train_tcf, y_train_tcf)

# determine r2 score for model
score_tcf = model_tcf.score(X_test_tcf, y_test_tcf)
display('R2 for TCF: ' + str(score_tcf))

# predict output for test data
predictions_tcf = model_tcf.predict(X_test_tcf)
predictionsDF_tcf = pd.DataFrame(predictions_tcf, columns=output_tcf)
```

```
# Plot parity of actual versus predicted values for flue temperature
parity_tcf = go.Figure()

# add test v. predicted markers
parity_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=predictionsDF_tcf['tcf'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=y_test_tcf['tcf'],
        name='parity'
    )
)

# update layout and title
parity_tcf.update_layout(height=800, width=800, title="Flue Temperature Actual vs. Predicted")
parity_tcf.update_xaxes(title='Actual TCF (°C)')
parity_tcf.update_yaxes(title='Predicted TCF (°C)')

# display figure
parity_tcf.show()
```

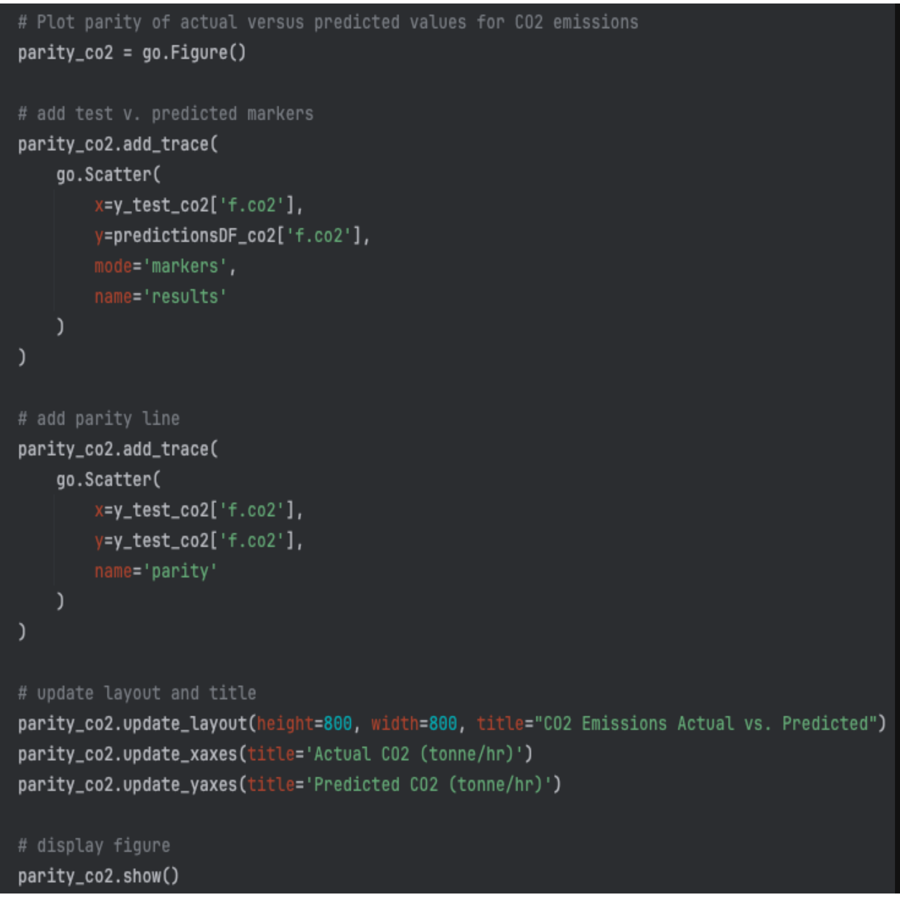


'R2 for TCF: 0.7611991641990963'

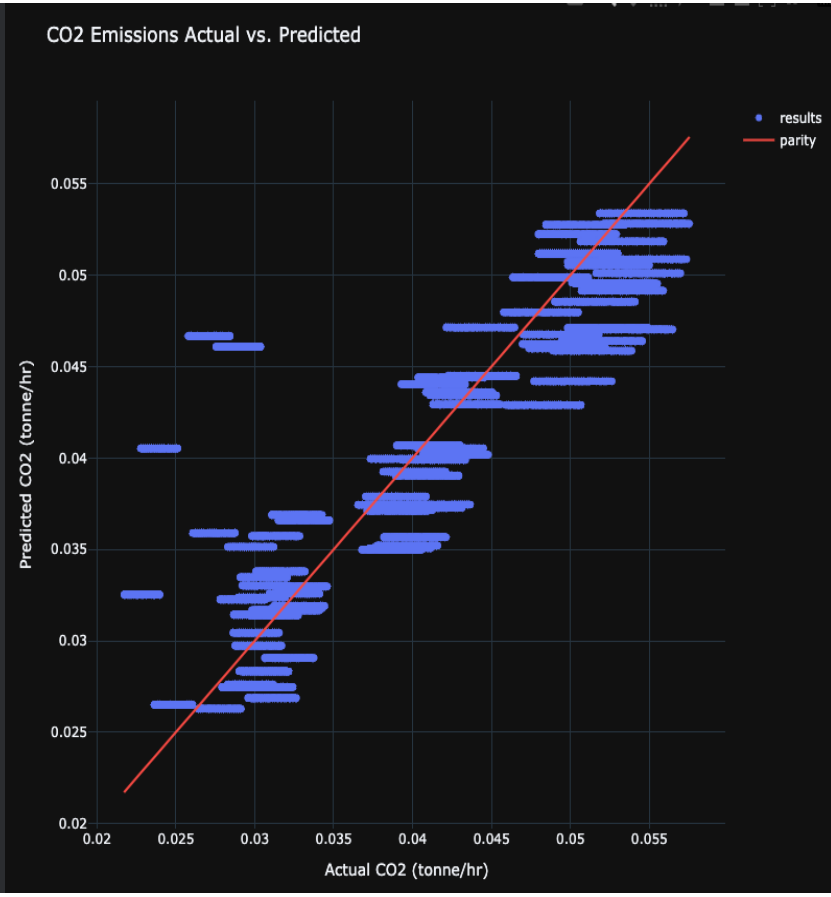
Predict CO2 emissions



```
# Build linear regression model for CO2 emissions (f.co2)
output_co2 = ['f.co2']

# split the data into training (80%) and testing (20%) sets
X_train_co2, X_test_co2, y_train_co2, y_test_co2 = train_test_split(
    df[inputs],
    df[output_co2],
    test_size=0.2,
    random_state=42
)

# initiate the linear regression model
model_co2 = LinearRegression()

# fit the linear model using the input data
model_co2.fit(X_train_co2, y_train_co2)

# determine r2 score for model
score_co2 = model_co2.score(X_test_co2, y_test_co2)
display('R2 for CO2: ' + str(score_co2))

# predict output for test data
predictions_co2 = model_co2.predict(X_test_co2)
predictionsDF_co2 = pd.DataFrame(predictions_co2, columns=output_co2)
```

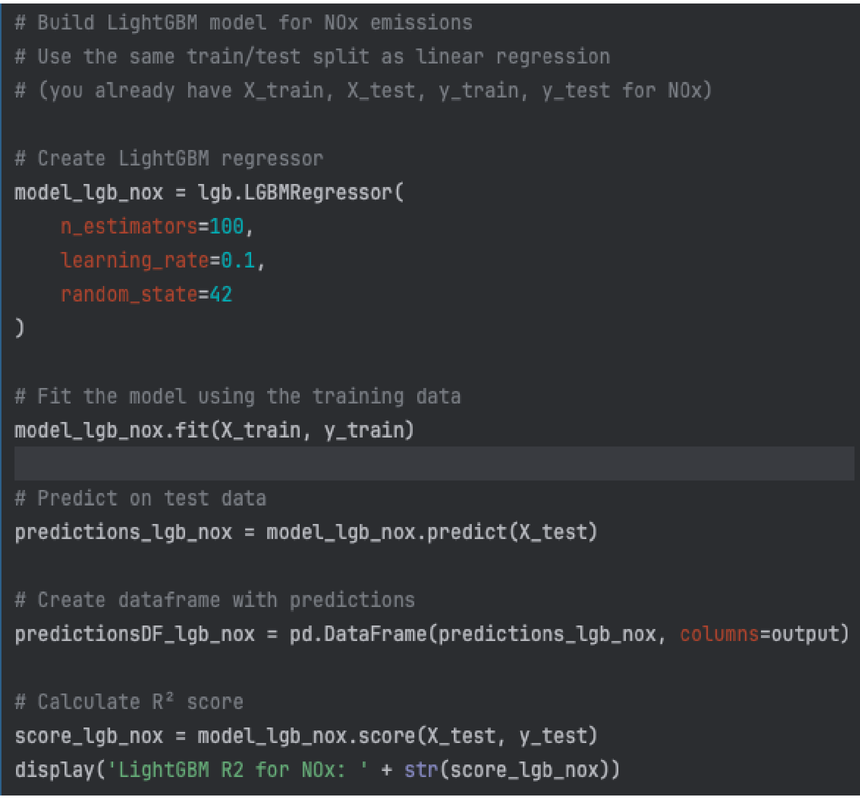


```
# Plot parity of actual versus predicted values for CO2 emissions
parity_co2 = go.Figure()

# add test v. predicted markers
parity_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=predictionsDF_co2['f.co2'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=y_test_co2['f.co2'],
        name='parity'
    )
)

# update layout and title
parity_co2.update_layout(height=800, width=800, title="CO2 Emissions Actual vs. Predicted")
parity_co2.update_xaxes(title='Actual CO2 (tonne/hr)')
parity_co2.update_yaxes(title='Predicted CO2 (tonne/hr)')

# display figure
parity_co2.show()
```

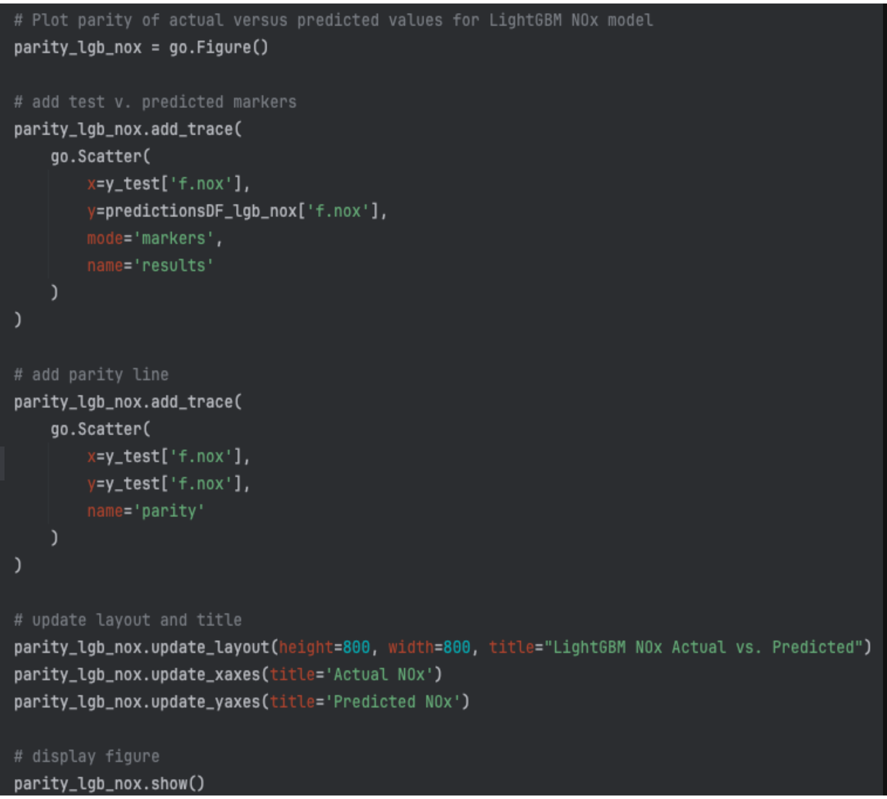


'R2 for CO2: 0.7714021644644824'

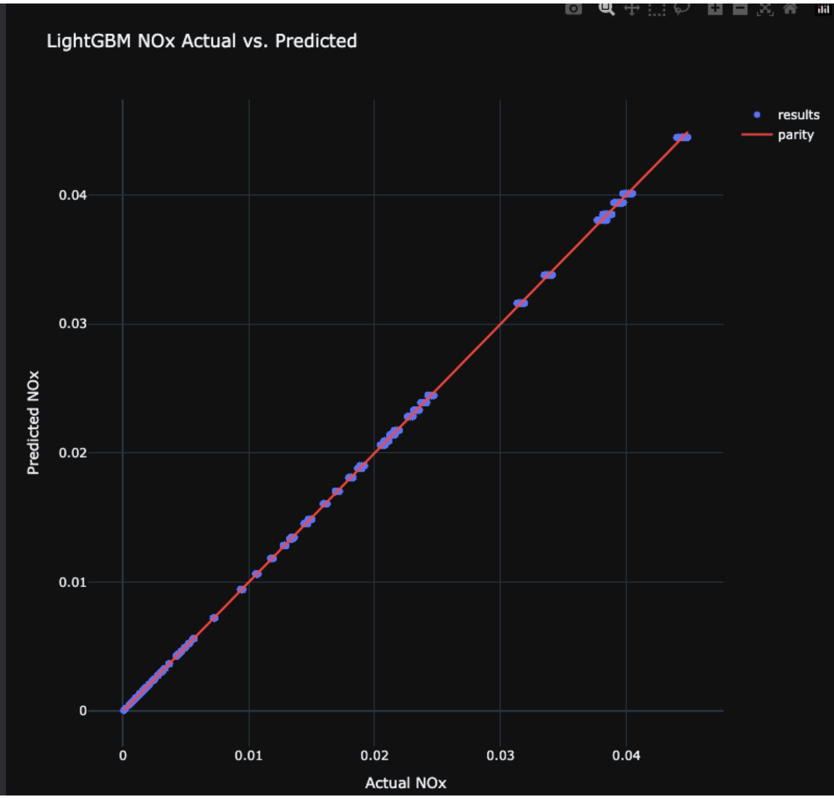
Task 4 – LightGBM - NOx emissions



```
# Build LightGBM model for NOx emissions
# Use the same train/test split as linear regression
# (you already have X_train, X_test, y_train, y_test for NOx)

# Create LightGBM regressor
model_lgb_nox = lgb.LGBMRegressor(
    n_estimators=100,
    learning_rate=0.1,
    random_state=42
)

# Fit the model using the training data
model_lgb_nox.fit(X_train, y_train)

# Predict on test data
predictions_lgb_nox = model_lgb_nox.predict(X_test)

# Create dataframe with predictions
predictionsDF_lgb_nox = pd.DataFrame(predictions_lgb_nox, columns=output)

# Calculate R² score
score_lgb_nox = model_lgb_nox.score(X_test, y_test)
display('LightGBM R2 for NOx: ' + str(score_lgb_nox))
```

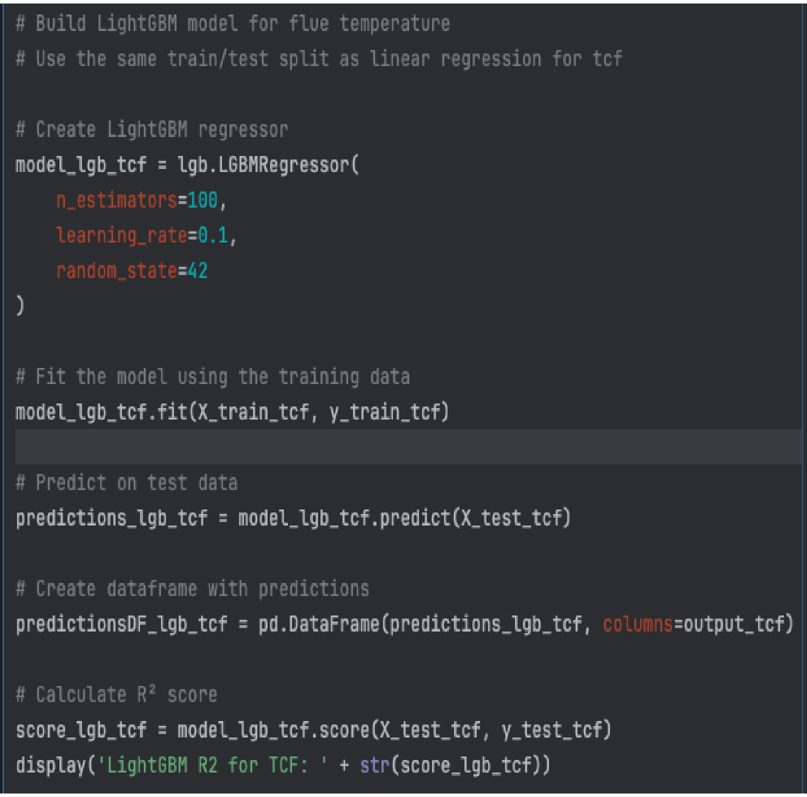


```
# Plot parity of actual versus predicted values for LightGBM NOx model
parity_lgb_nox = go.Figure()

# add test v. predicted markers
parity_lgb_nox.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=predictionsDF_lgb_nox['f.nox'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity_lgb_nox.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=y_test['f.nox'],
        name='parity'
    )
)

# update layout and title
parity_lgb_nox.update_layout(height=800, width=800, title="LightGBM NOx Actual vs. Predicted")
parity_lgb_nox.update_xaxes(title='Actual NOx')
parity_lgb_nox.update_yaxes(title='Predicted NOx')

# display figure
parity_lgb_nox.show()
```

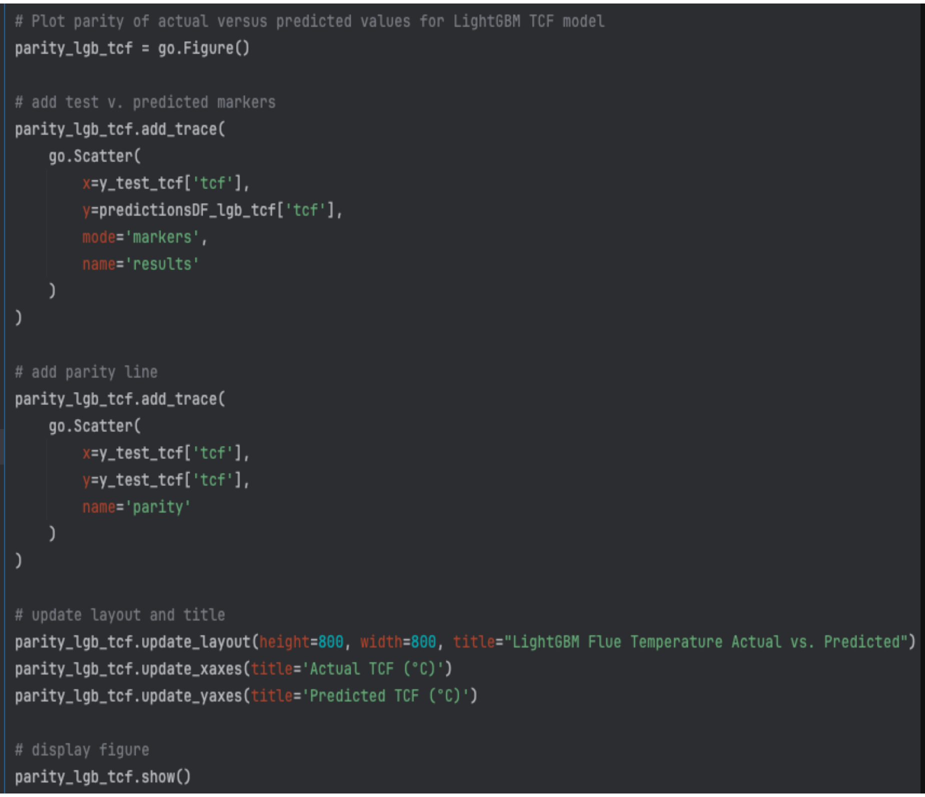


'LightGBM R2 for NOx: 0.9999476812166448'

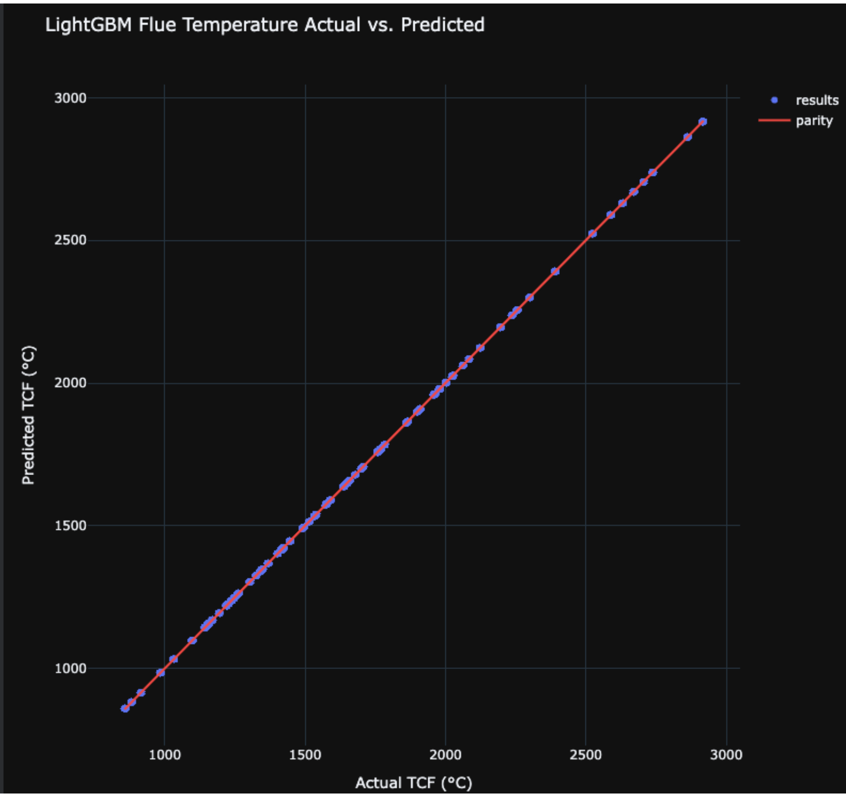
Predict Flue temperature



```python
# Build LightGBM model for flue temperature
# Use the same train/test split as linear regression for tcf

# Create LightGBM regressor
model_lgb_tcf = lgb.LGBMRegressor(
    n_estimators=100,
    learning_rate=0.1,
    random_state=42
)

# Fit the model using the training data
model_lgb_tcf.fit(X_train_tcf, y_train_tcf)

# Predict on test data
predictions_lgb_tcf = model_lgb_tcf.predict(X_test_tcf)

# Create dataframe with predictions
predictionsDF_lgb_tcf = pd.DataFrame(predictions_lgb_tcf, columns=output_tcf)

# Calculate R² score
score_lgb_tcf = model_lgb_tcf.score(X_test_tcf, y_test_tcf)
display('LightGBM R2 for TCF: ' + str(score_lgb_tcf))
```

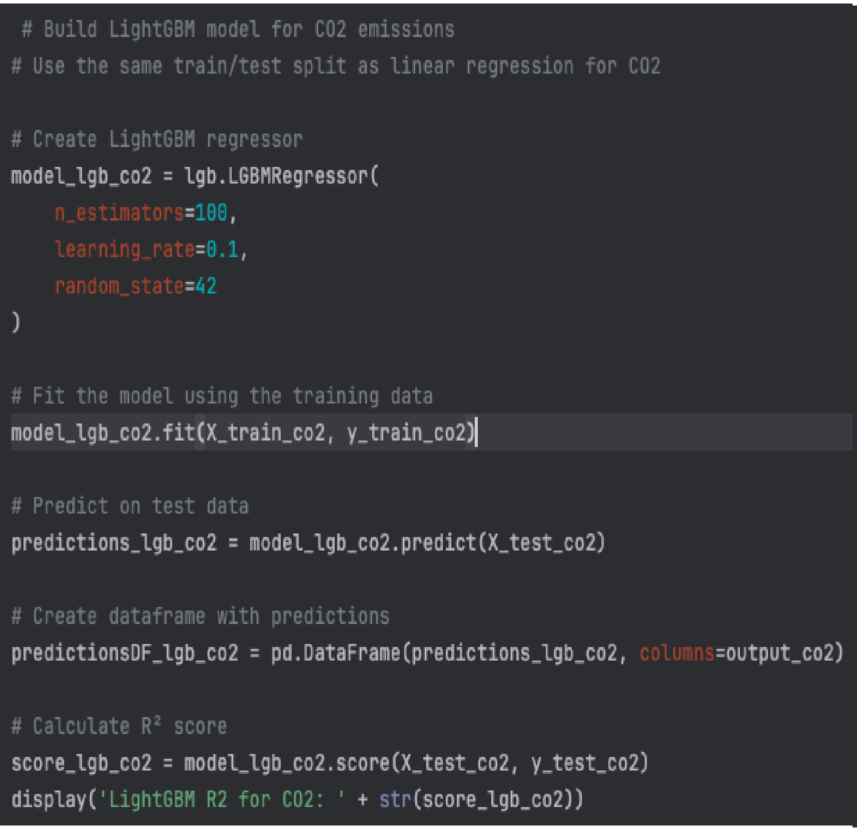


```python
# Plot parity of actual versus predicted values for LightGBM TCF model
parity_lgb_tcf = go.Figure()

# add test v. predicted markers
parity_lgb_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=predictionsDF_lgb_tcf['tcf'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity_lgb_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=y_test_tcf['tcf'],
        name='parity'
    )
)

# update layout and title
parity_lgb_tcf.update_layout(height=800, width=800, title="LightGBM Flue Temperature Actual vs. Predicted")
parity_lgb_tcf.update_xaxes(title='Actual TCF (°C)')
parity_lgb_tcf.update_yaxes(title='Predicted TCF (°C)')

# display figure
parity_lgb_tcf.show()
```

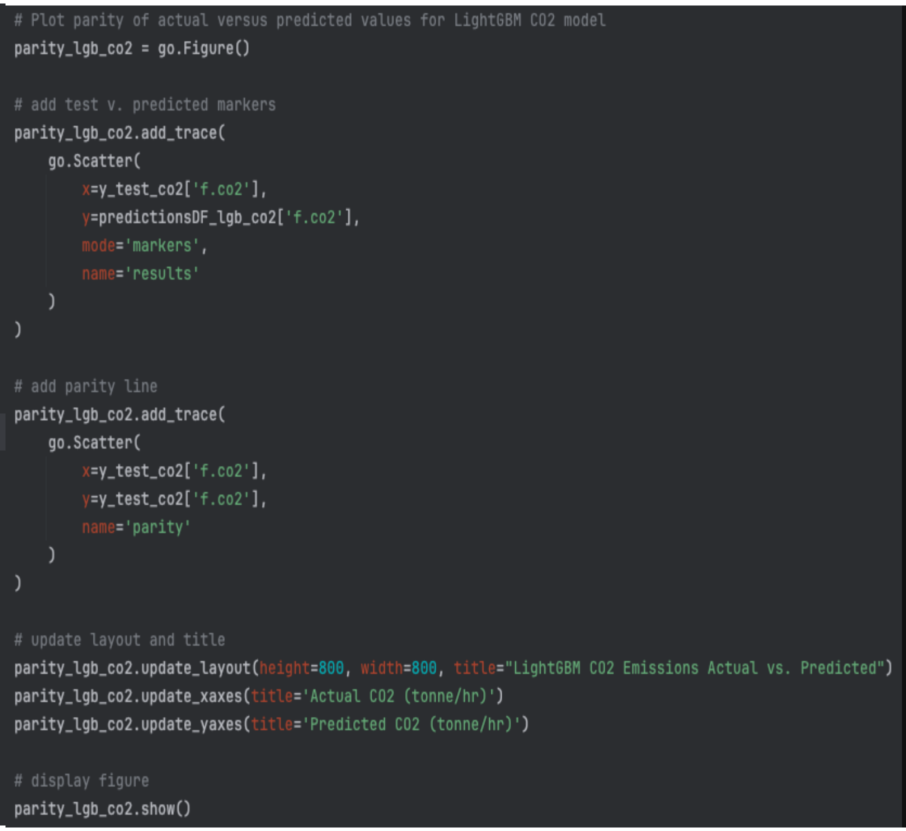


'LightGBM R2 for TCF: 0.9999863375702582'

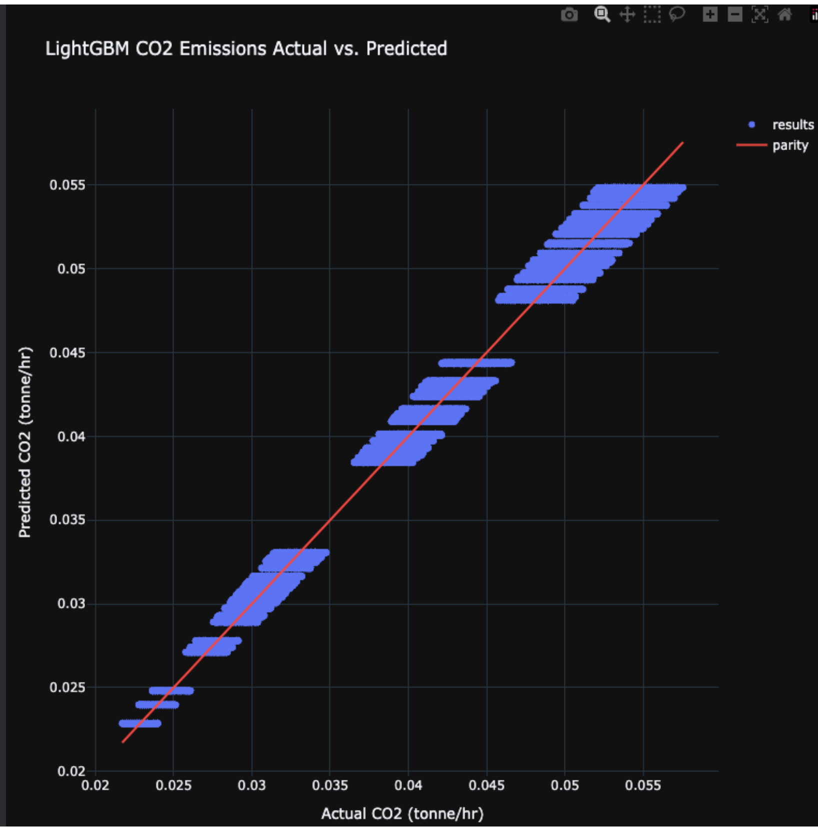
Predict CO2 Emissions



```
 # Build LightGBM model for CO2 emissions
# Use the same train/test split as linear regression for CO2

# Create LightGBM regressor
model_lgb_co2 = lgb.LGBMRegressor(
    n_estimators=100,
    learning_rate=0.1,
    random_state=42
)

# Fit the model using the training data
model_lgb_co2.fit(X_train_co2, y_train_co2)

# Predict on test data
predictions_lgb_co2 = model_lgb_co2.predict(X_test_co2)

# Create dataframe with predictions
predictionsDF_lgb_co2 = pd.DataFrame(predictions_lgb_co2, columns=output_co2)

# Calculate R² score
score_lgb_co2 = model_lgb_co2.score(X_test_co2, y_test_co2)
display('LightGBM R2 for CO2: ' + str(score_lgb_co2))
```

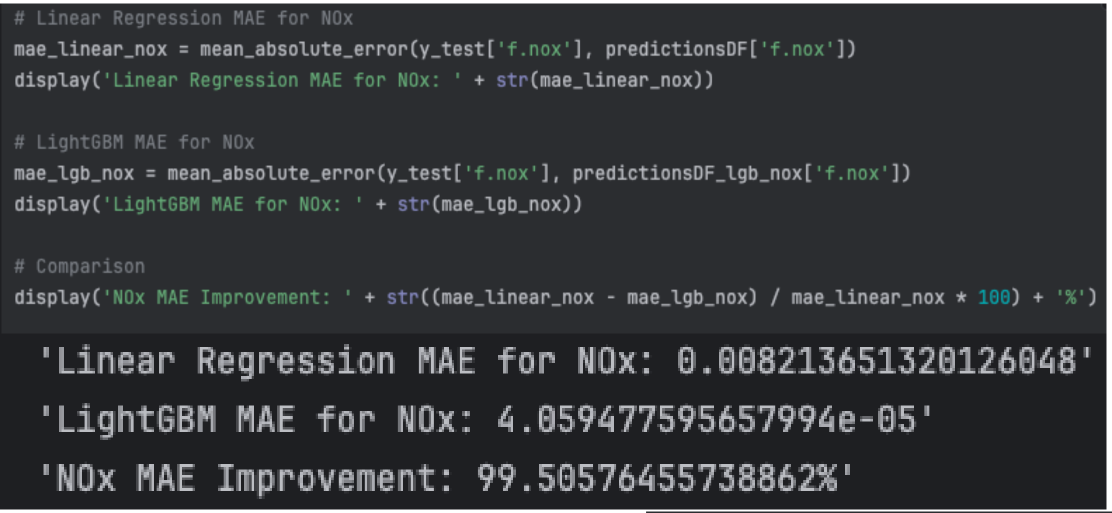


```
# Plot parity of actual versus predicted values for LightGBM CO2 model
parity_lgb_co2 = go.Figure()

# add test v. predicted markers
parity_lgb_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=predictionsDF_lgb_co2['f.co2'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity_lgb_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=y_test_co2['f.co2'],
        name='parity'
    )
)

# update layout and title
parity_lgb_co2.update_layout(height=800, width=800, title="LightGBM CO2 Emissions Actual vs. Predicted")
parity_lgb_co2.update_xaxes(title='Actual CO2 (tonne/hr)')
parity_lgb_co2.update_yaxes(title='Predicted CO2 (tonne/hr)')

# display figure
parity_lgb_co2.show()
```



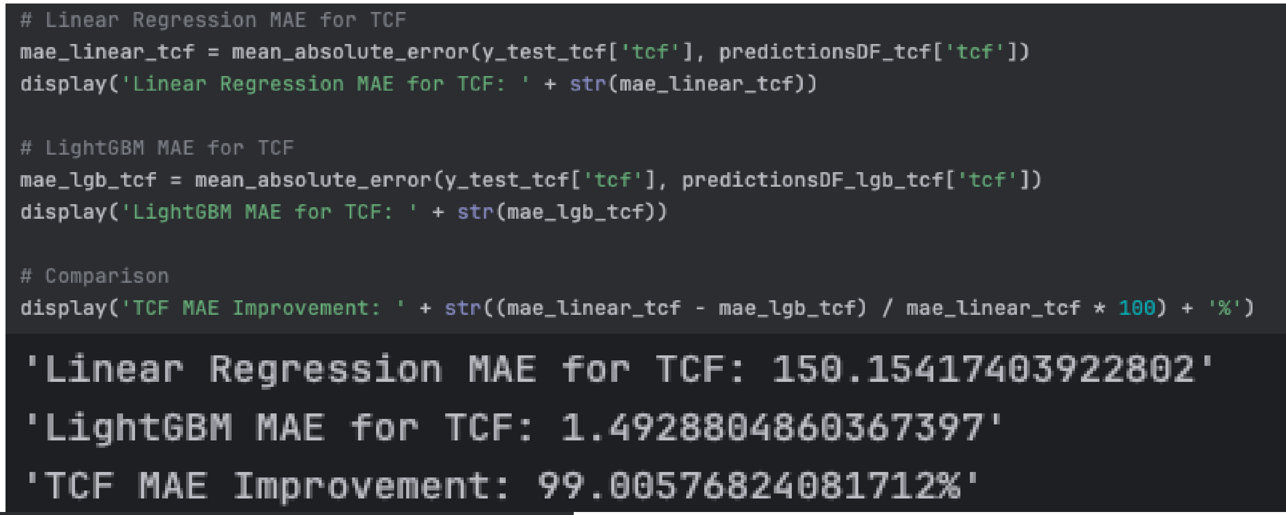
'LightGBM R2 for CO2: 0.9833328919378389'

MAE Calculation



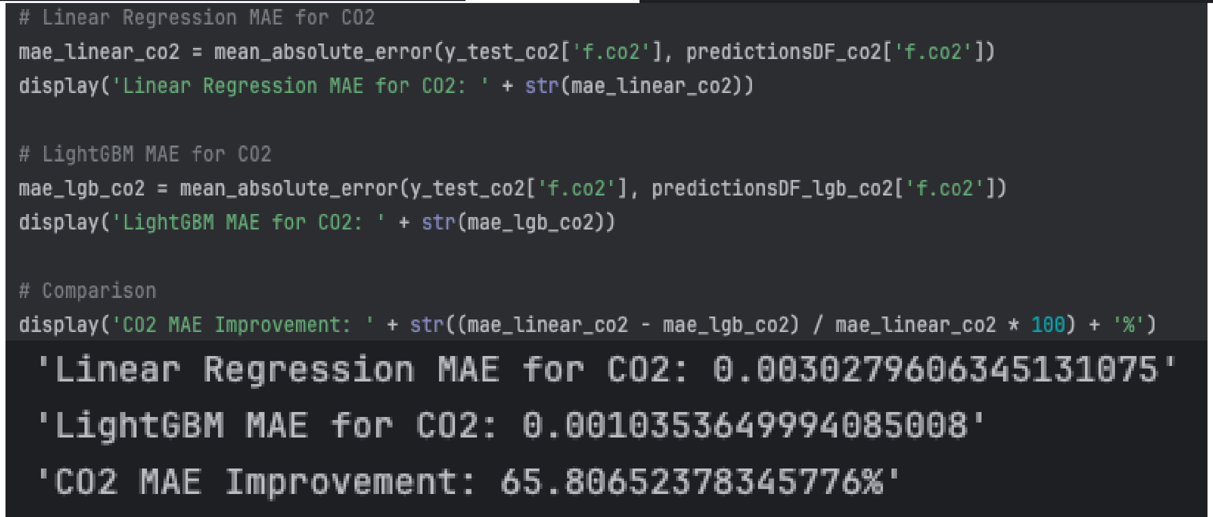
```
# Linear Regression MAE for NOx
mae_linear_nox = mean_absolute_error(y_test['f.nox'], predictionsDF['f.nox'])
display('Linear Regression MAE for NOx: ' + str(mae_linear_nox))

# LightGBM MAE for NOx
mae_lgb_nox = mean_absolute_error(y_test['f.nox'], predictionsDF_lgb_nox['f.nox'])
display('LightGBM MAE for NOx: ' + str(mae_lgb_nox))

# Comparison
display('NOx MAE Improvement: ' + str((mae_linear_nox - mae_lgb_nox) / mae_linear_nox * 100) + '%')
```

```
'Linear Regression MAE for NOx: 0.008213651320126048'
'LightGBM MAE for NOx: 4.059477595657994e-05'
'NOx MAE Improvement: 99.50576455738862%'
```



```
# Linear Regression MAE for TCF
mae_linear_tcf = mean_absolute_error(y_test_tcf['tcf'], predictionsDF_tcf['tcf'])
display('Linear Regression MAE for TCF: ' + str(mae_linear_tcf))

# LightGBM MAE for TCF
mae_lgb_tcf = mean_absolute_error(y_test_tcf['tcf'], predictionsDF_lgb_tcf['tcf'])
display('LightGBM MAE for TCF: ' + str(mae_lgb_tcf))

# Comparison
display('TCF MAE Improvement: ' + str((mae_linear_tcf - mae_lgb_tcf) / mae_linear_tcf * 100) + '%')
```

```
'Linear Regression MAE for TCF: 150.15417403922802'
'LightGBM MAE for TCF: 1.4928804860367397'
'TCF MAE Improvement: 99.00576824081712%'
```



```
# Linear Regression MAE for CO2
mae_linear_co2 = mean_absolute_error(y_test_co2['f.co2'], predictionsDF_co2['f.co2'])
display('Linear Regression MAE for CO2: ' + str(mae_linear_co2))

# LightGBM MAE for CO2
mae_lgb_co2 = mean_absolute_error(y_test_co2['f.co2'], predictionsDF_lgb_co2['f.co2'])
display('LightGBM MAE for CO2: ' + str(mae_lgb_co2))

# Comparison
display('CO2 MAE Improvement: ' + str((mae_linear_co2 - mae_lgb_co2) / mae_linear_co2 * 100) + '%')
```

```
'Linear Regression MAE for CO2: 0.0030279606345131075'
'LightGBM MAE for CO2: 0.0010353649994085008'
'CO2 MAE Improvement: 65.80652378345776%'
```

R2 & MAE Comparison

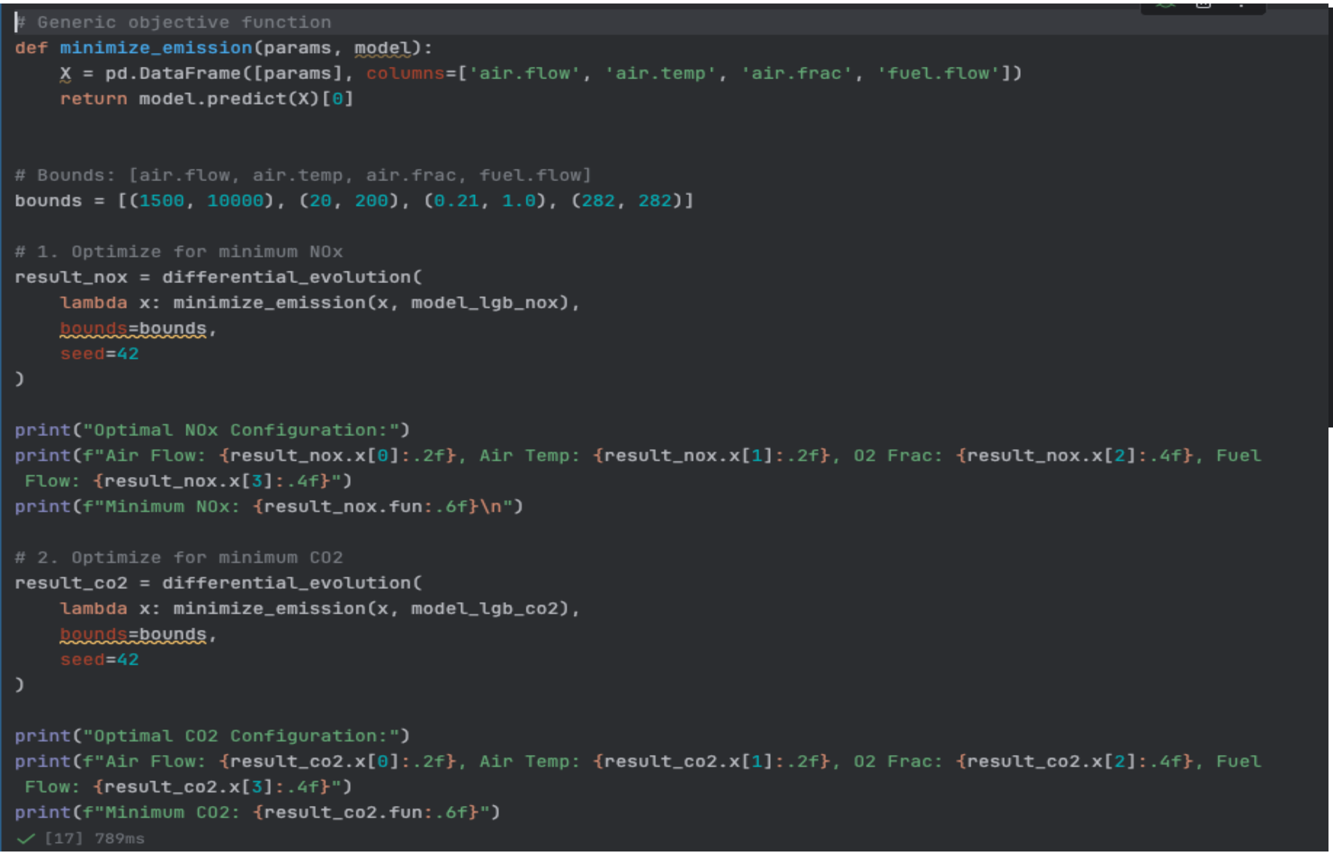
Performance Comparison

Model	Linear R2	LightGBM R2	Improvement
NOx	0.099	0.999	Yes
Temperature	0.761	0.999	Yes
CO2	0.771	0.983	Yes

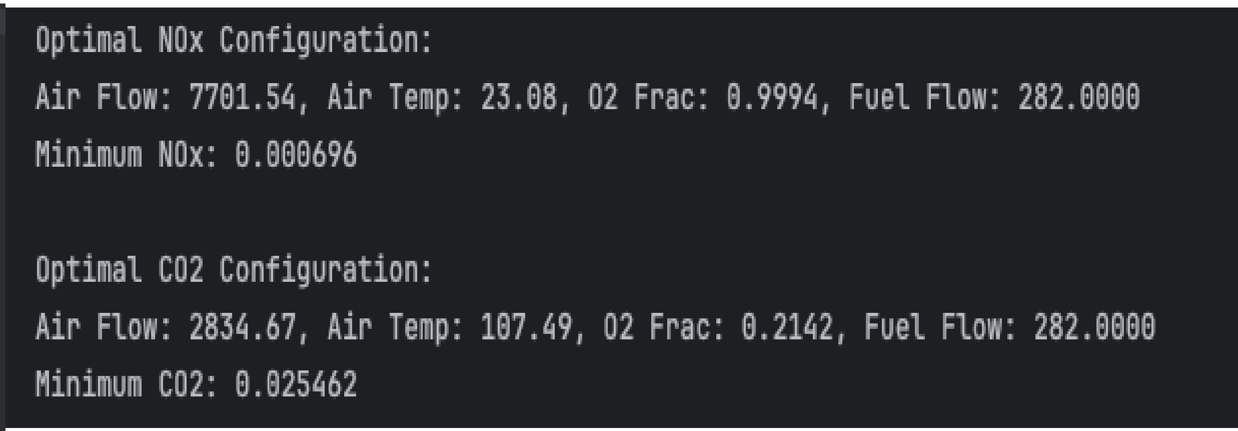
Mean Absolute Error

Model	Linear MAE	LightGBM MAE	Reduction
NOx	0.00821	0.0000406	99.5%
Temperature	150.15	1.49	99.0%
CO2	0.00303	0.00104	65.8%

Optimization

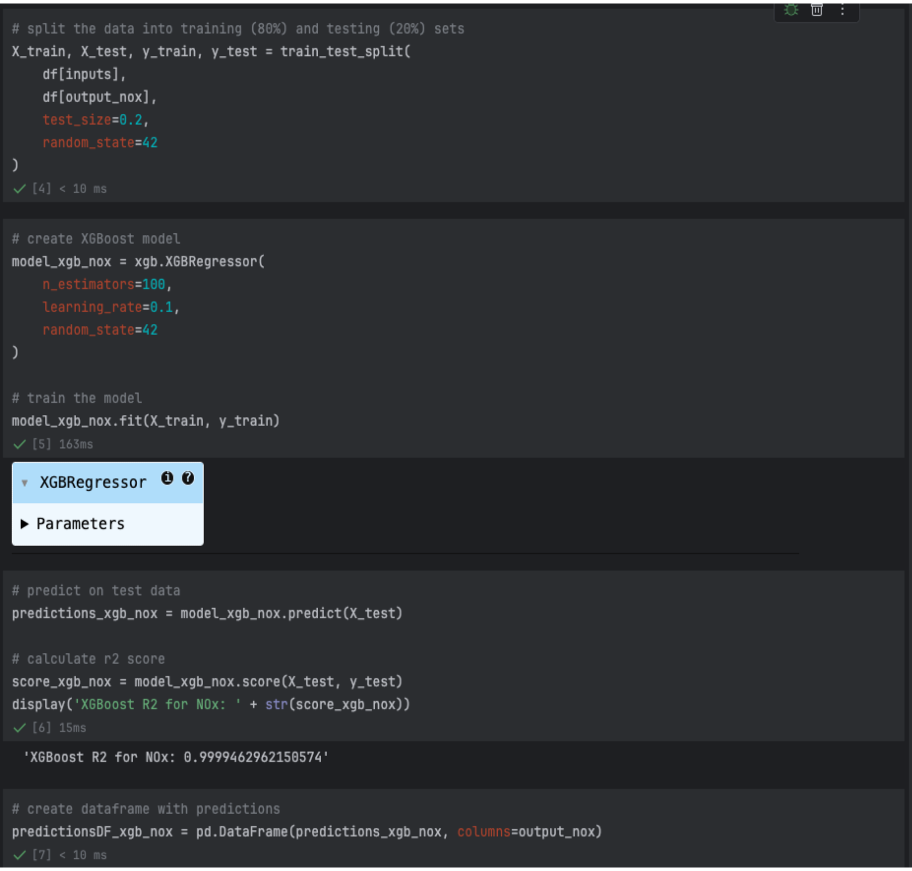


```
# Generic objective function
def minimize_emission(params, model):
    X = pd.DataFrame([params], columns=['air.flow', 'air.temp', 'air.frac', 'fuel.flow'])
    return model.predict(X)[0]


# Bounds: [air.flow, air.temp, air.frac, fuel.flow]
bounds = [(1500, 10000), (20, 200), (0.21, 1.0), (282, 282)]

# 1. Optimize for minimum NOx
result_nox = differential_evolution(
    lambda x: minimize_emission(x, model_lgb_nox),
    bounds=bounds,
    seed=42
)

print("Optimal NOx Configuration:")
print(f"Air Flow: {result_nox.x[0]:.2f}, Air Temp: {result_nox.x[1]:.2f}, O2 Frac: {result_nox.x[2]:.4f}, Fuel
 Flow: {result_nox.x[3]:.4f}")
print(f"Minimum NOx: {result_nox.fun:.6f}\n")

# 2. Optimize for minimum CO2
result_co2 = differential_evolution(
    lambda x: minimize_emission(x, model_lgb_co2),
    bounds=bounds,
    seed=42
)

print("Optimal CO2 Configuration:")
print(f"Air Flow: {result_co2.x[0]:.2f}, Air Temp: {result_co2.x[1]:.2f}, O2 Frac: {result_co2.x[2]:.4f}, Fuel
 Flow: {result_co2.x[3]:.4f}")
print(f"Minimum CO2: {result_co2.fun:.6f}")
✓ [17] 789ms
```



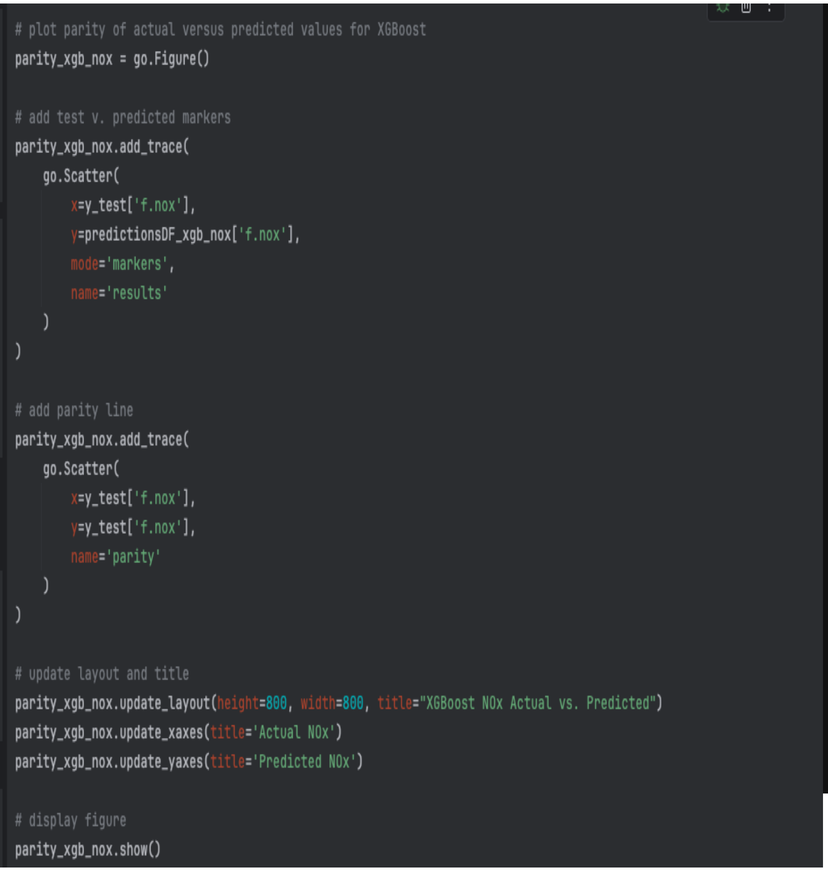
```
Optimal NOx Configuration:
Air Flow: 7701.54, Air Temp: 23.08, O2 Frac: 0.9994, Fuel Flow: 282.0000
Minimum NOx: 0.000696

Optimal CO2 Configuration:
Air Flow: 2834.67, Air Temp: 107.49, O2 Frac: 0.2142, Fuel Flow: 282.0000
Minimum CO2: 0.025462
```

XGBoost for NOX

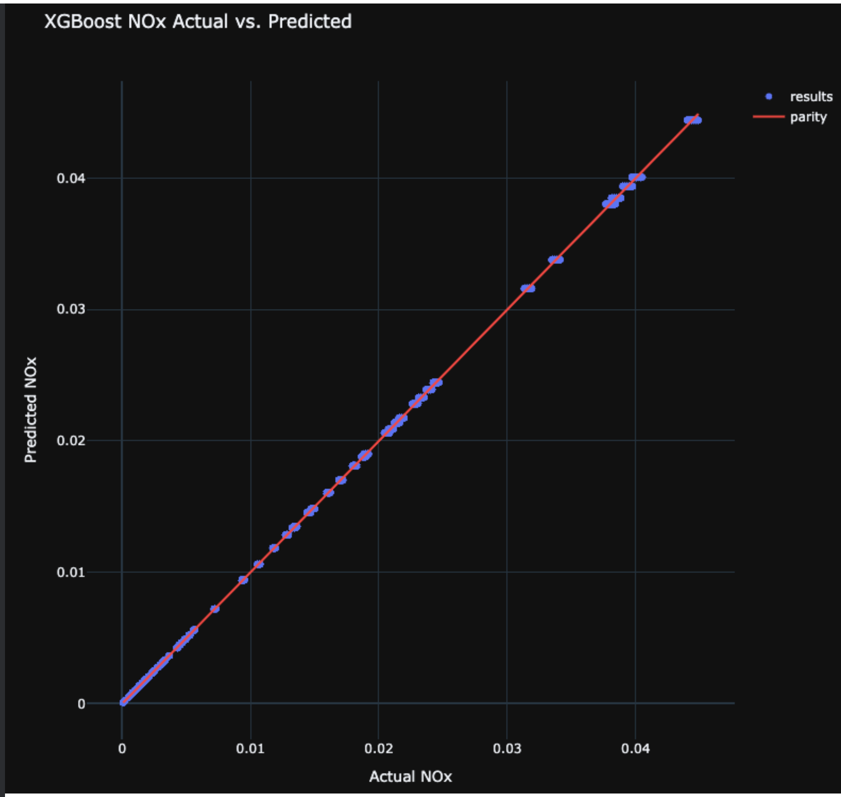


```
# split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(
    df[inputs],
    df[output_nox],
    test_size=0.2,
    random_state=42
)
✓ [4] < 10 ms

# create XGBoost model
model_xgb_nox = xgb.XGBRegressor(
    n_estimators=100,
    learning_rate=0.1,
    random_state=42
)

# train the model
model_xgb_nox.fit(X_train, y_train)
✓ [5] 163ms
```

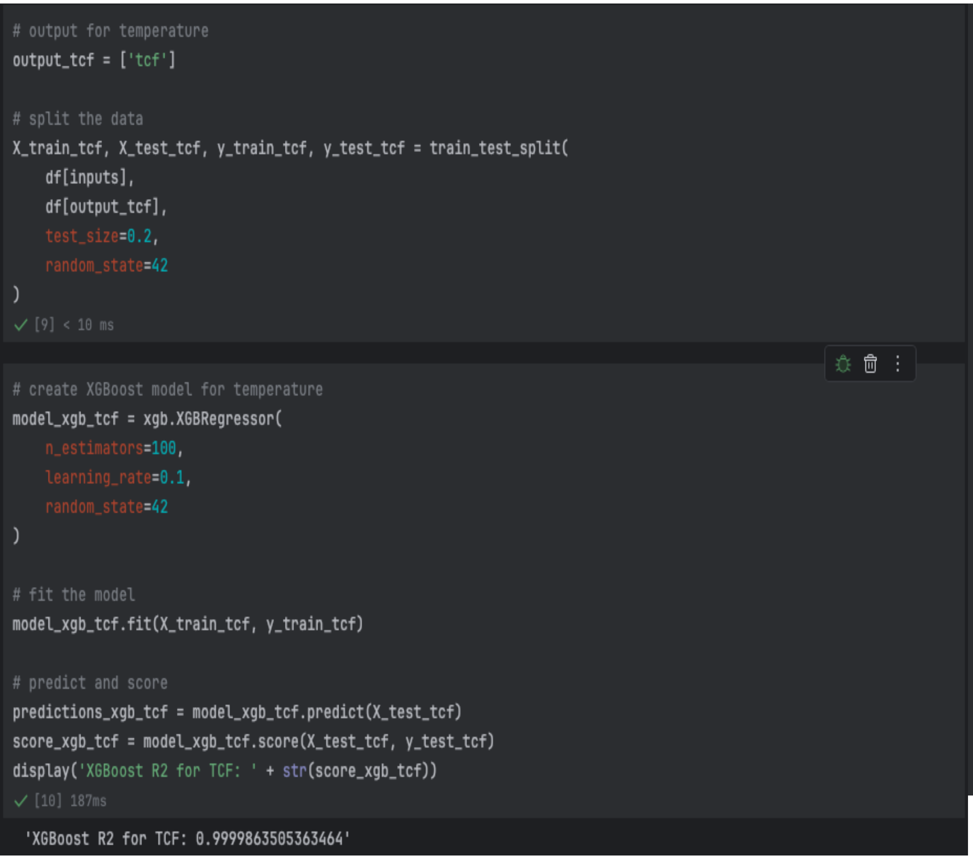
```
# predict on test data
predictions_xgb_nox = model_xgb_nox.predict(X_test)

# calculate r2 score
score_xgb_nox = model_xgb_nox.score(X_test, y_test)
display('XGBoost R2 for NOx: ' + str(score_xgb_nox))
✓ [6] 15ms

 'XGBoost R2 for NOx: 0.9999462962150574'

# create dataframe with predictions
predictionsDF_xgb_nox = pd.DataFrame(predictions_xgb_nox, columns=output_nox)
✓ [7] < 10 ms
```



```
# plot parity of actual versus predicted values for XGBoost
parity_xgb_nox = go.Figure()

# add test v. predicted markers
parity_xgb_nox.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=predictionsDF_xgb_nox['f.nox'],
        mode='markers',
        name='results'
    )
)

# add parity line
parity_xgb_nox.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=y_test['f.nox'],
        name='parity'
    )
)

# update layout and title
parity_xgb_nox.update_layout(height=800, width=800, title="XGBoost NOx Actual vs. Predicted")
parity_xgb_nox.update_xaxes(title='Actual NOx')
parity_xgb_nox.update_yaxes(title='Predicted NOx')

# display figure
parity_xgb_nox.show()
```

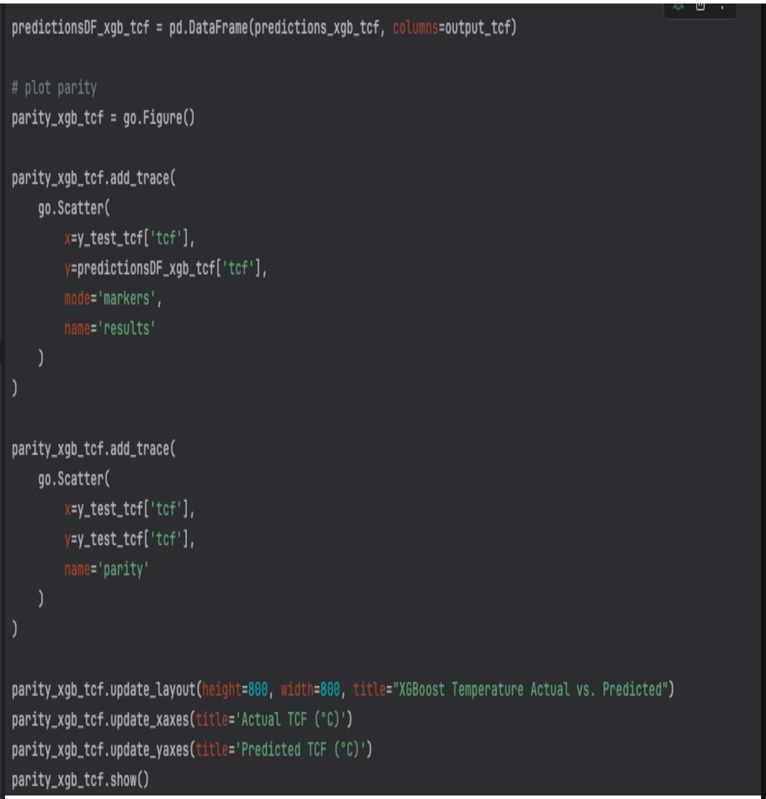


XGBoost for Temperature

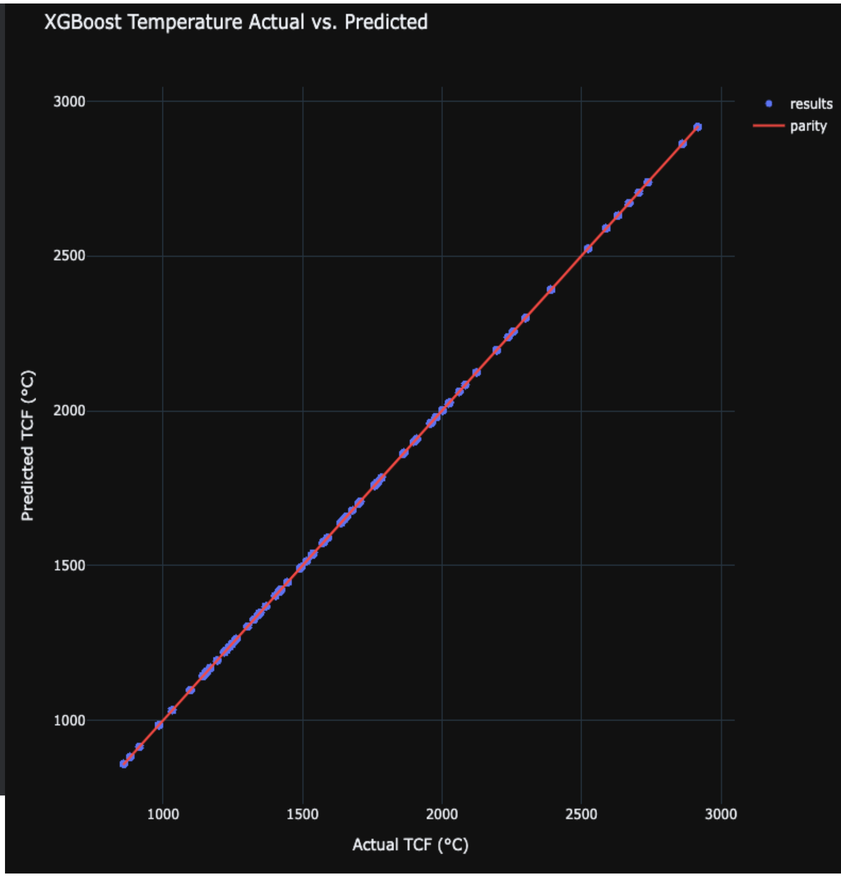


```
# output for temperature
output_tcf = ['tcf']

# split the data
X_train_tcf, X_test_tcf, y_train_tcf, y_test_tcf = train_test_split(
    df[inputs],
    df[output_tcf],
    test_size=0.2,
    random_state=42
)
✓ [9] < 10 ms
```

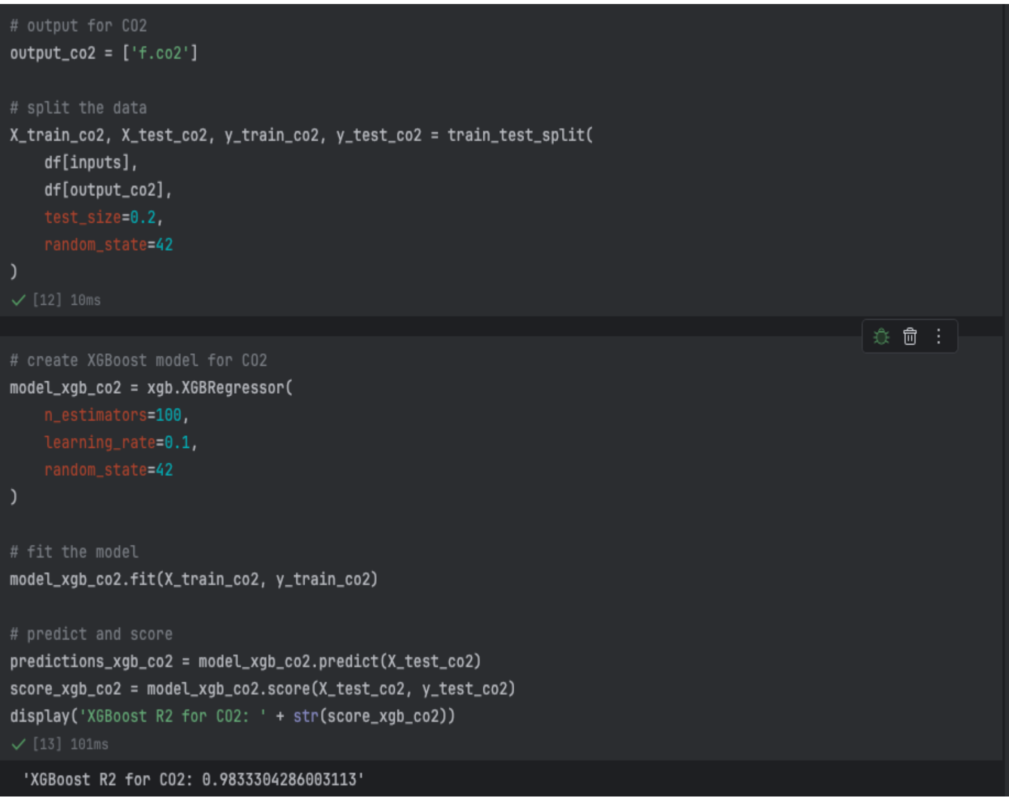
```
# create XGBoost model for temperature
model_xgb_tcf = xgb.XGBRegressor(
    n_estimators=100,
    learning_rate=0.1,
    random_state=42
)

# fit the model
model_xgb_tcf.fit(X_train_tcf, y_train_tcf)

# predict and score
predictions_xgb_tcf = model_xgb_tcf.predict(X_test_tcf)
score_xgb_tcf = model_xgb_tcf.score(X_test_tcf, y_test_tcf)
display('XGBoost R2 for TCF: ' + str(score_xgb_tcf))
✓ [10] 187ms

 'XGBoost R2 for TCF: 0.9999863505363464'
```



```
predictionsDF_xgb_tcf = pd.DataFrame(predictions_xgb_tcf, columns=output_tcf)

# plot parity
parity_xgb_tcf = go.Figure()

parity_xgb_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=predictionsDF_xgb_tcf['tcf'],
        mode='markers',
        name='results'
    )
)

parity_xgb_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=y_test_tcf['tcf'],
        name='parity'
    )
)

parity_xgb_tcf.update_layout(height=800, width=800, title="XGBoost Temperature Actual vs. Predicted")
parity_xgb_tcf.update_xaxes(title='Actual TCF (°C)')
parity_xgb_tcf.update_yaxes(title='Predicted TCF (°C)')
parity_xgb_tcf.show()
```

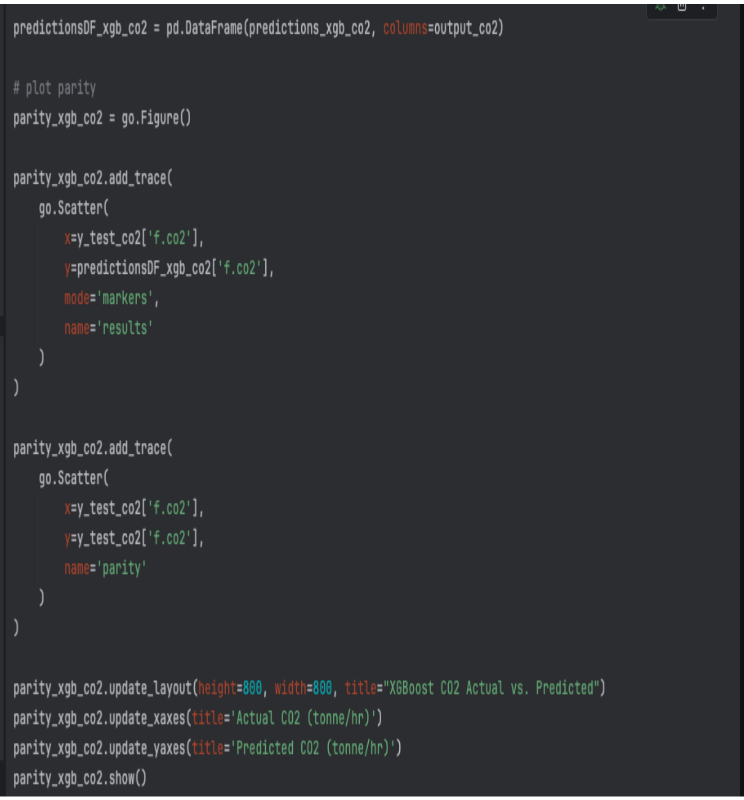


XGBoost for Co2

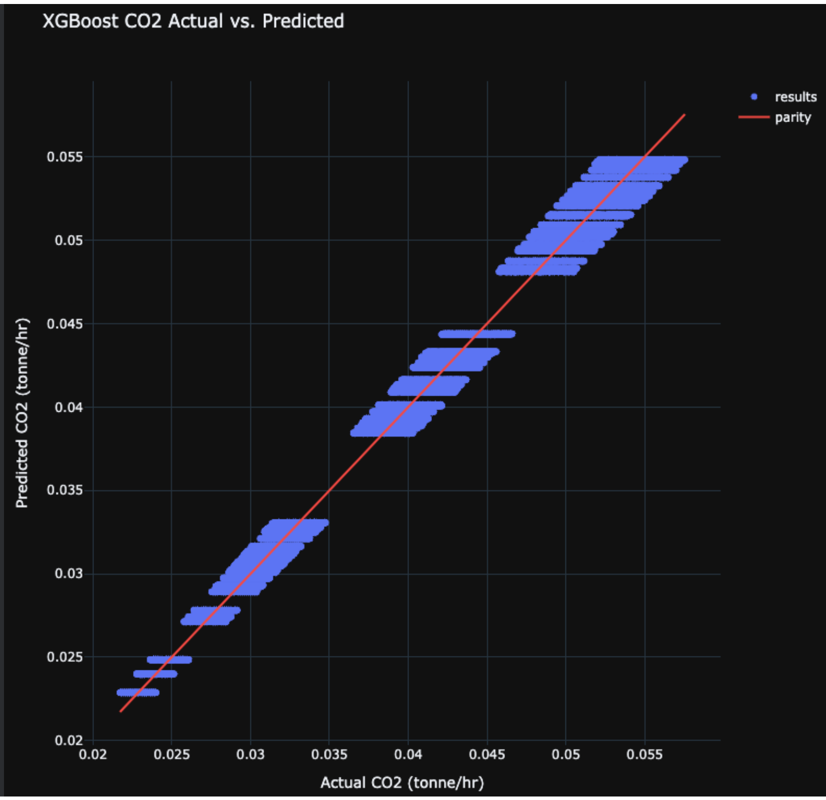


```
# output for CO2
output_co2 = ['f.co2']

# split the data
X_train_co2, X_test_co2, y_train_co2, y_test_co2 = train_test_split(
    df[inputs],
    df[output_co2],
    test_size=0.2,
    random_state=42
)
✓ [12] 10ms
```

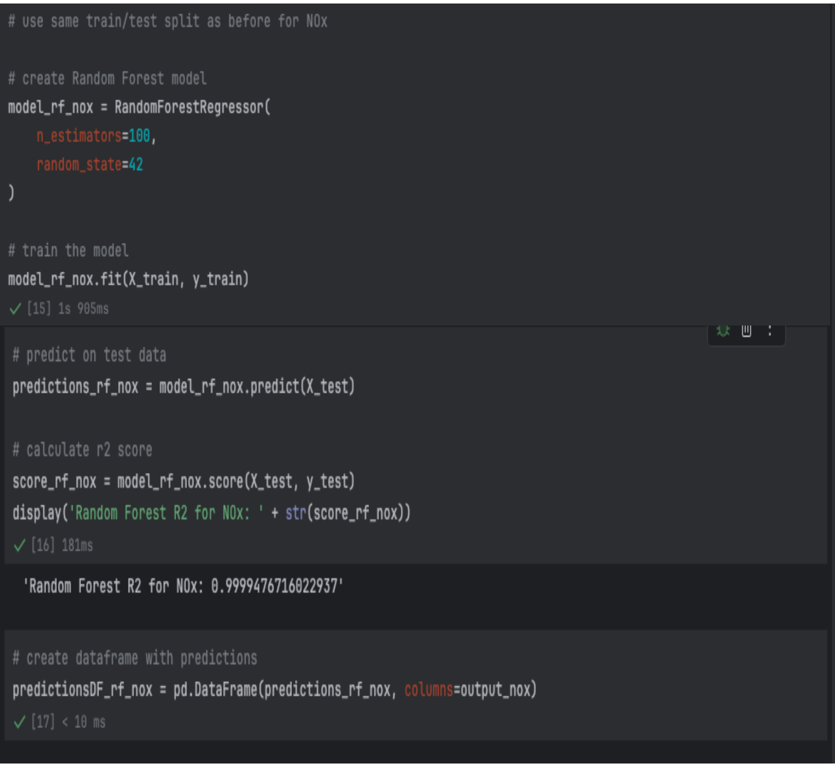
```
# create XGBoost model for CO2
model_xgb_co2 = xgb.XGBRegressor(
    n_estimators=100,
    learning_rate=0.1,
    random_state=42
)

# fit the model
model_xgb_co2.fit(X_train_co2, y_train_co2)

# predict and score
predictions_xgb_co2 = model_xgb_co2.predict(X_test_co2)
score_xgb_co2 = model_xgb_co2.score(X_test_co2, y_test_co2)
display('XGBoost R2 for CO2: ' + str(score_xgb_co2))
✓ [13] 101ms
  'XGBoost R2 for CO2: 0.9833304286003113'
```



```
predictionsDF_xgb_co2 = pd.DataFrame(predictions_xgb_co2, columns=output_co2)

# plot parity
parity_xgb_co2 = go.Figure()

parity_xgb_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=predictionsDF_xgb_co2['f.co2'],
        mode='markers',
        name='results'
    )
)

parity_xgb_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=y_test_co2['f.co2'],
        name='parity'
    )
)

parity_xgb_co2.update_layout(height=800, width=800, title="XGBoost CO2 Actual vs. Predicted")
parity_xgb_co2.update_xaxes(title='Actual CO2 (tonne/hr)')
parity_xgb_co2.update_yaxes(title='Predicted CO2 (tonne/hr)')
parity_xgb_co2.show()
```



Random Forest for NOX

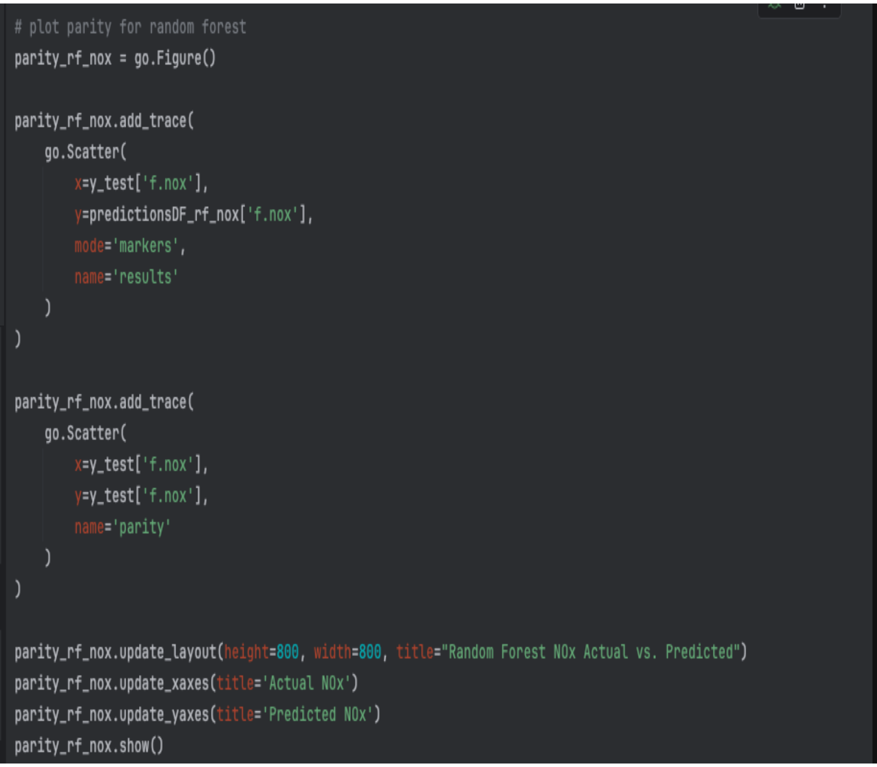


```
# use same train/test split as before for NOx

# create Random Forest model
model_rf_nox = RandomForestRegressor(
    n_estimators=100,
    random_state=42
)

# train the model
model_rf_nox.fit(X_train, y_train)
✓ [15] 1s 905ms
```

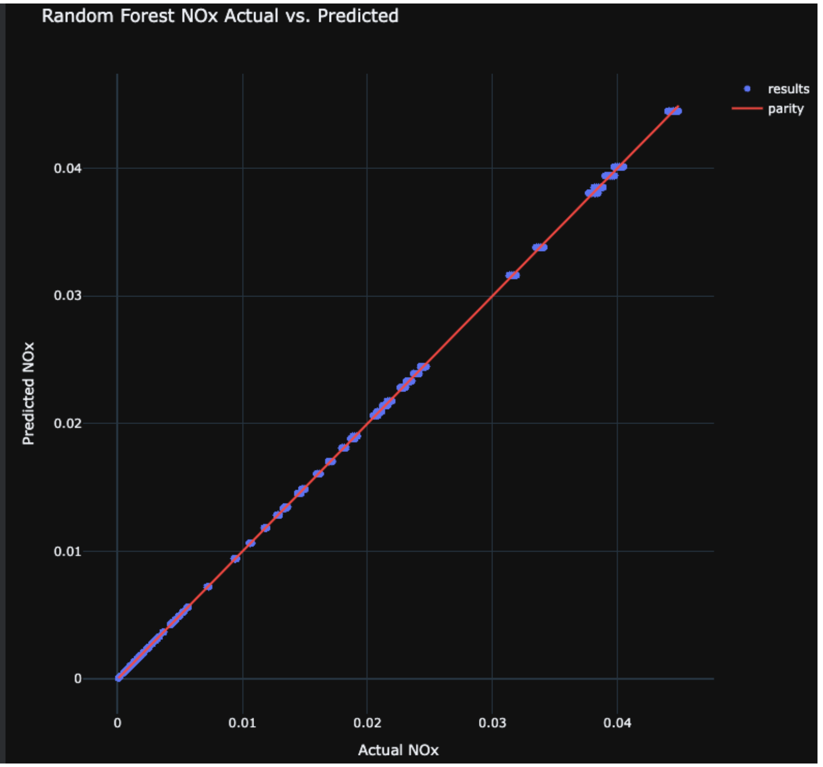
```
# predict on test data
predictions_rf_nox = model_rf_nox.predict(X_test)

# calculate r2 score
score_rf_nox = model_rf_nox.score(X_test, y_test)
display('Random Forest R2 for NOx: ' + str(score_rf_nox))
✓ [16] 181ms
```

```
'Random Forest R2 for NOx: 0.9999476716022937'
```

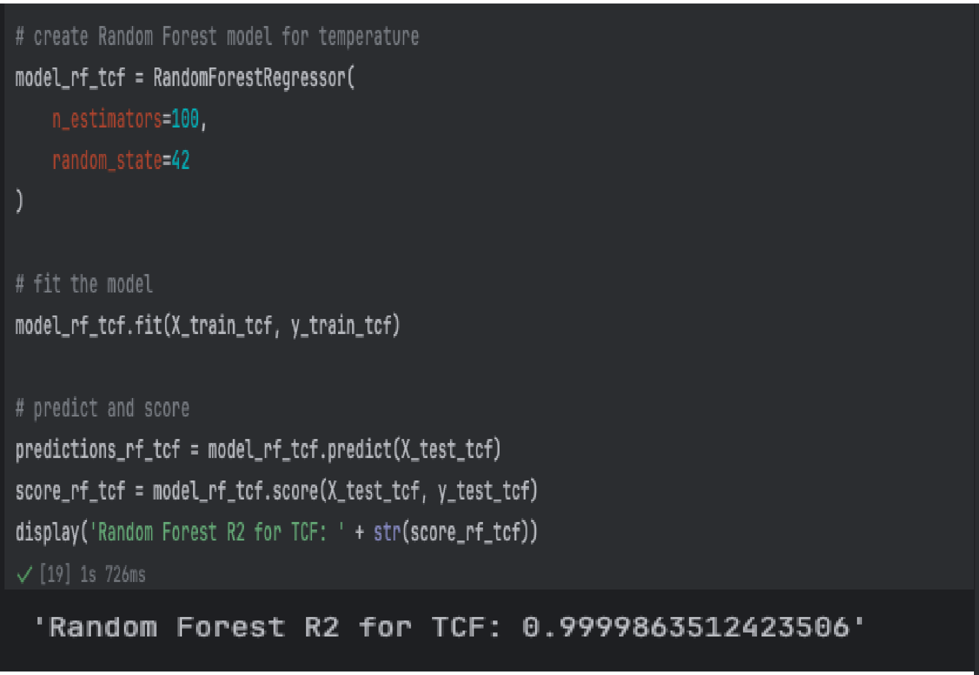
```
# create dataframe with predictions
predictionsDF_rf_nox = pd.DataFrame(predictions_rf_nox, columns=output_nox)
✓ [17] < 10 ms
```



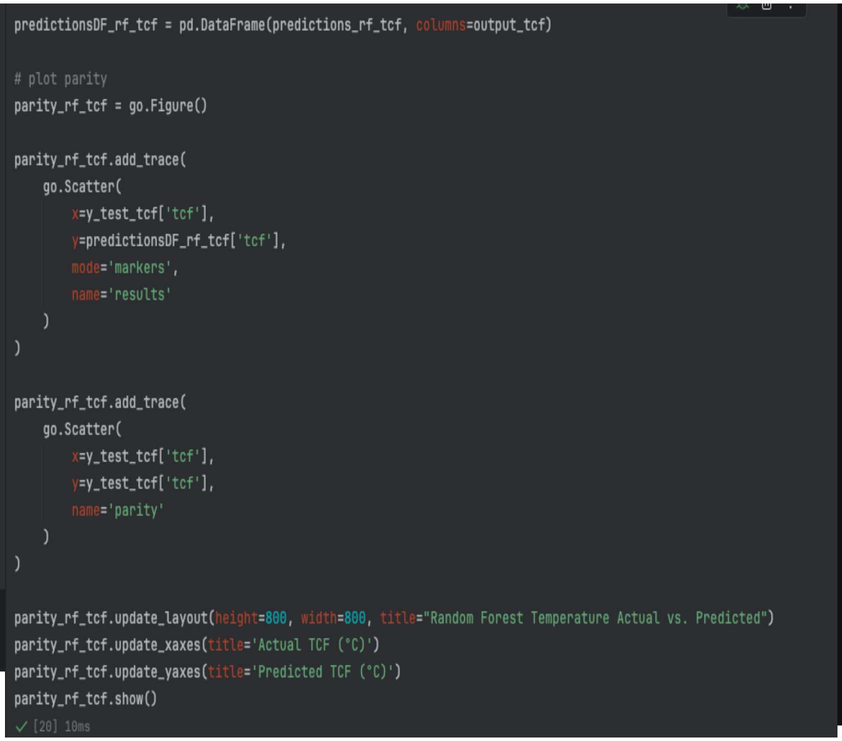
```
# plot parity for random forest
parity_rf_nox = go.Figure()

parity_rf_nox.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=predictionsDF_rf_nox['f.nox'],
        mode='markers',
        name='results'
    )
)

parity_rf_nox.add_trace(
    go.Scatter(
        x=y_test['f.nox'],
        y=y_test['f.nox'],
        name='parity'
    )
)

parity_rf_nox.update_layout(height=800, width=800, title="Random Forest NOx Actual vs. Predicted")
parity_rf_nox.update_xaxes(title='Actual NOx')
parity_rf_nox.update_yaxes(title='Predicted NOx')
parity_rf_nox.show()
```



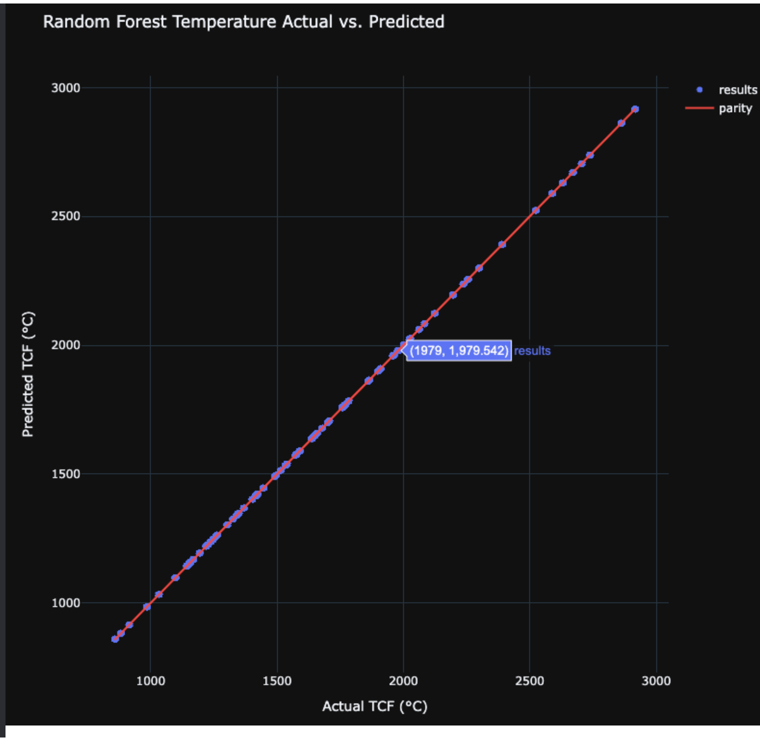
Random Forest for Temperature



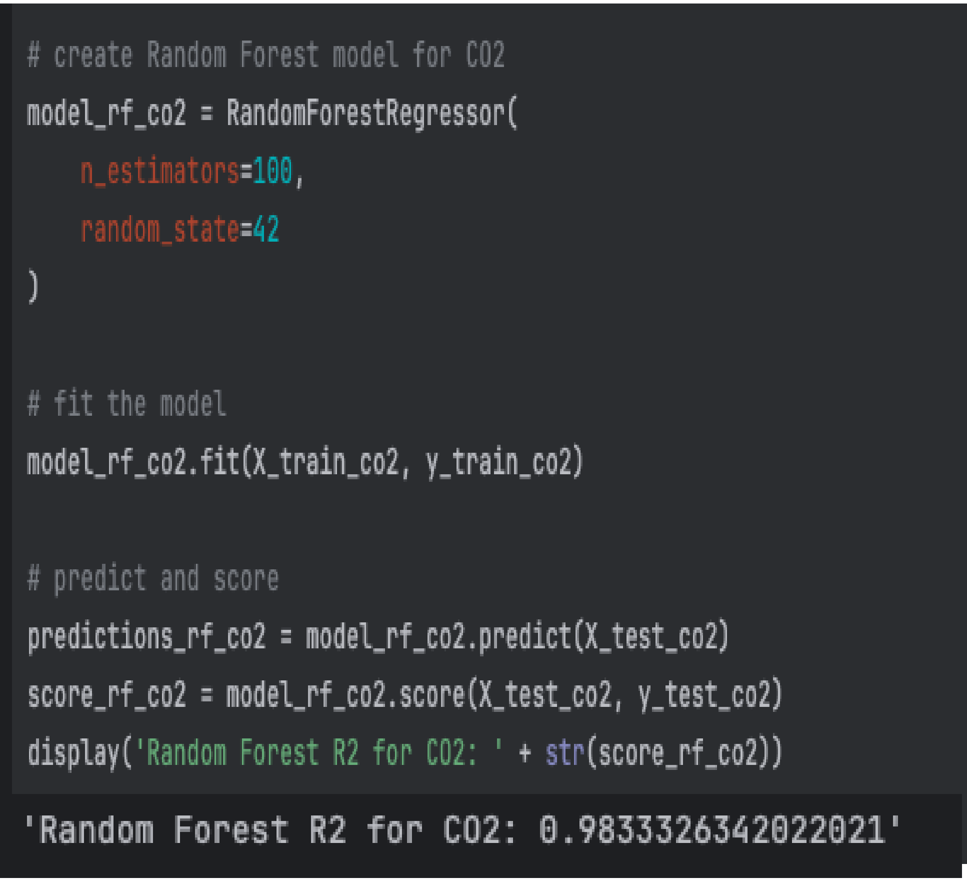
```
# create Random Forest model for temperature
model_rf_tcf = RandomForestRegressor(
    n_estimators=100,
    random_state=42
)

# fit the model
model_rf_tcf.fit(X_train_tcf, y_train_tcf)

# predict and score
predictions_rf_tcf = model_rf_tcf.predict(X_test_tcf)
score_rf_tcf = model_rf_tcf.score(X_test_tcf, y_test_tcf)
display('Random Forest R2 for TCF: ' + str(score_rf_tcf))
✓ [19] 1s 726ms

'Random Forest R2 for TCF: 0.9999863512423506'
```



```
predictionsDF_rf_tcf = pd.DataFrame(predictions_rf_tcf, columns=output_tcf)

# plot parity
parity_rf_tcf = go.Figure()

parity_rf_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=predictionsDF_rf_tcf['tcf'],
        mode='markers',
        name='results'
    )
)

parity_rf_tcf.add_trace(
    go.Scatter(
        x=y_test_tcf['tcf'],
        y=y_test_tcf['tcf'],
        name='parity'
    )
)

parity_rf_tcf.update_layout(height=800, width=800, title="Random Forest Temperature Actual vs. Predicted")
parity_rf_tcf.update_xaxes(title='Actual TCF (°C)')
parity_rf_tcf.update_yaxes(title='Predicted TCF (°C)')
parity_rf_tcf.show()
✓ [20] 10ms
```

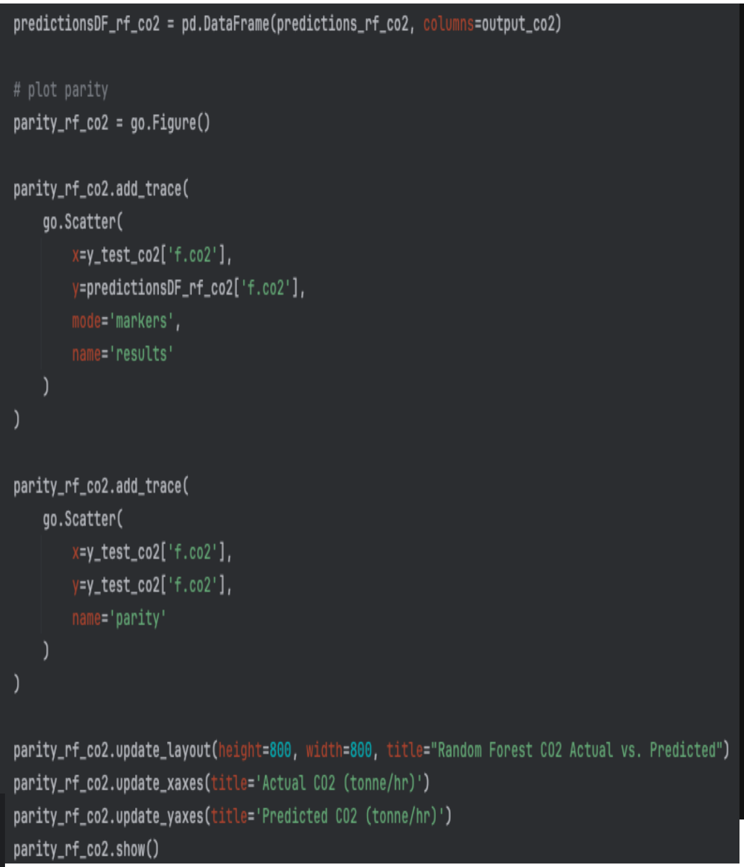


Random Forest for Co2

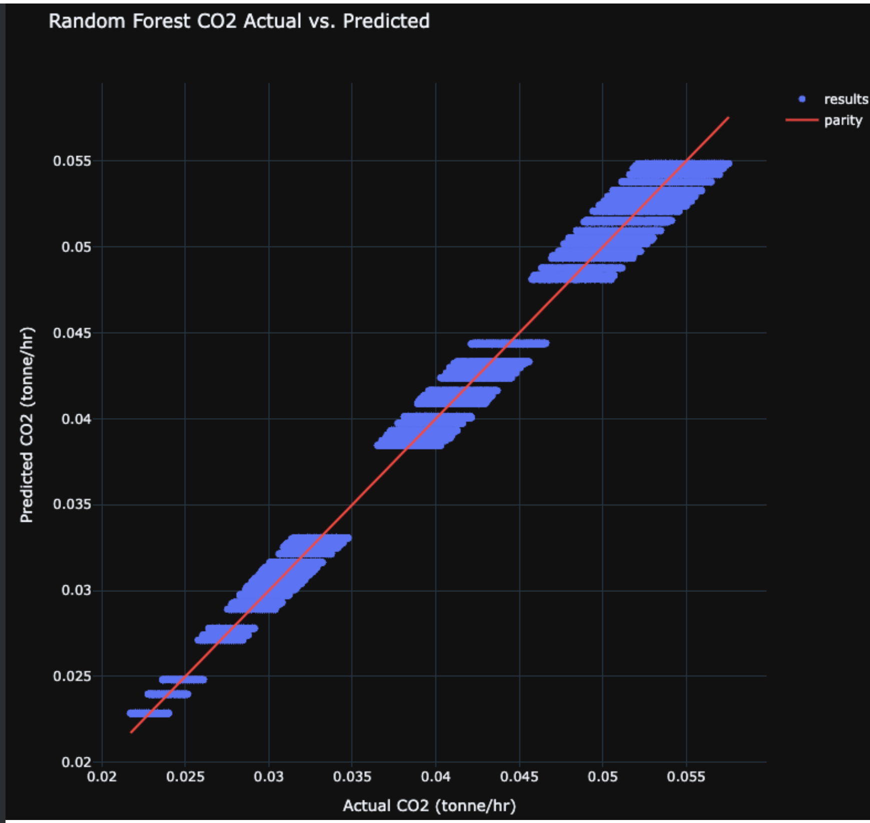


```
# create Random Forest model for CO2
model_rf_co2 = RandomForestRegressor(
    n_estimators=100,
    random_state=42
)

# fit the model
model_rf_co2.fit(X_train_co2, y_train_co2)

# predict and score
predictions_rf_co2 = model_rf_co2.predict(X_test_co2)
score_rf_co2 = model_rf_co2.score(X_test_co2, y_test_co2)
display('Random Forest R2 for CO2: ' + str(score_rf_co2))
```

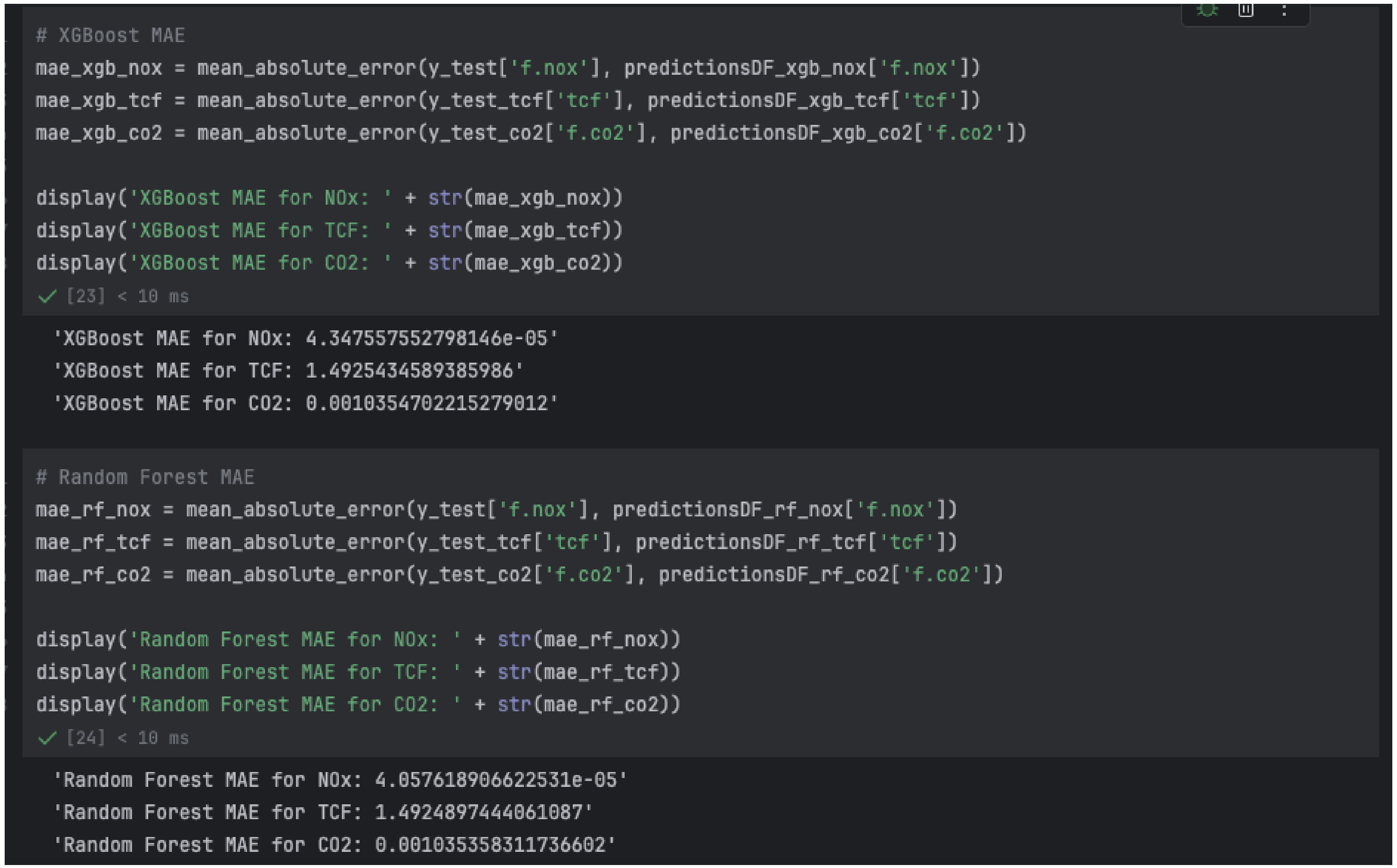
```
'Random Forest R2 for CO2: 0.9833263420220 21'
```



```
predictionsDF_rf_co2 = pd.DataFrame(predictions_rf_co2, columns=output_co2)

# plot parity
parity_rf_co2 = go.Figure()

parity_rf_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=predictionsDF_rf_co2['f.co2'],
        mode='markers',
        name='results'
    )
)

parity_rf_co2.add_trace(
    go.Scatter(
        x=y_test_co2['f.co2'],
        y=y_test_co2['f.co2'],
        name='parity'
    )
)

parity_rf_co2.update_layout(height=800, width=800, title="Random Forest CO2 Actual vs. Predicted")
parity_rf_co2.update_xaxes(title='Actual CO2 (tonne/hr)')
parity_rf_co2.update_yaxes(title='Predicted CO2 (tonne/hr)')
parity_rf_co2.show()
```



MAE Comparison



```
# XGBoost MAE
mae_xgb_nox = mean_absolute_error(y_test['f.nox'], predictionsDF_xgb_nox['f.nox'])
mae_xgb_tcf = mean_absolute_error(y_test_tcf['tcf'], predictionsDF_xgb_tcf['tcf'])
mae_xgb_co2 = mean_absolute_error(y_test_co2['f.co2'], predictionsDF_xgb_co2['f.co2'])

display('XGBoost MAE for NOx: ' + str(mae_xgb_nox))
display('XGBoost MAE for TCF: ' + str(mae_xgb_tcf))
display('XGBoost MAE for CO2: ' + str(mae_xgb_co2))
```

✓ [23] < 10 ms

```
'XGBoost MAE for NOx: 4.347557552798146e-05'
'XGBoost MAE for TCF: 1.4925434589385986'
'XGBoost MAE for CO2: 0.0010354702215279012'
```

```
# Random Forest MAE
mae_rf_nox = mean_absolute_error(y_test['f.nox'], predictionsDF_rf_nox['f.nox'])
mae_rf_tcf = mean_absolute_error(y_test_tcf['tcf'], predictionsDF_rf_tcf['tcf'])
mae_rf_co2 = mean_absolute_error(y_test_co2['f.co2'], predictionsDF_rf_co2['f.co2'])

display('Random Forest MAE for NOx: ' + str(mae_rf_nox))
display('Random Forest MAE for TCF: ' + str(mae_rf_tcf))
display('Random Forest MAE for CO2: ' + str(mae_rf_co2))
```

✓ [24] < 10 ms

```
'Random Forest MAE for NOx: 4.057618906622531e-05'
'Random Forest MAE for TCF: 1.4924897444061087'
'Random Forest MAE for CO2: 0.001035358311736602'
```

Tested 4 Different Regression Models

Model	Type	Description
Linear Regression	Linear	Weighted Combination
LightGBM	Tree Based	Gradient boosting optimization
XGBoost	Tree Based	Extreme gradient boosting
Random Forest	Tree Based	Ensemble of decision Trees

NOX Prediction Results

Model	R^2 Scores	MAE
Linear Regression	0.0993	0.008214
LightGBM	0.9999	0.0000406
XGBoost	0.9999	0.0000470
Random Forest	0.9999	0.0000410

Temperature Prediction Results

Model	R^2	MAE
Linear Regression	0.7612	150.15°C
LightGBM	0.9999	1.493°C
XGBoost	0.9999	1.493°C
Random Forest	0.9999	1.492°C

Co2 Prediction Results

Model	R^2	MAE
Linear Regression	0.7714	0.003028
LightGBM	0.9833	0.001035
XGBoost	0.9833	0.001035
Random Forest	0.9833	0.001035