Formula 1

Business Problem: What factors/features are needed to predict race outcome

Things to do

- Outliers
- Natural Log

Powerpoint

- Agenda
- Background
- Business Problem: What factors/features are needed to predict race outcome
- Technical Apendix

1. Import required libraries

Get Data from the erghast F1 API Contains data from all the races since 1950

In [ ]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression

from statsmodels.formula.api import ols
from scipy.stats import norm, probplot
from sklearn.preprocessing import StandardScaler


db_dir = os.getcwd()
pit_stops = pd.read_csv(db_dir + r'/data/pit_stops.csv')
races = pd.read_csv(db_dir + r'/data/races.csv')
drivers = pd.read_csv(db_dir + r'/data/drivers.csv')
results = pd.read_csv(db_dir + r'/data/results.csv')
driver_standings = pd.read_csv(db_dir + r'/data/driver_standings.csv')
circuits = pd.read_csv(db_dir + r'/data/circuits.csv')
qualifying = pd.read_csv(db_dir + r'/data/qualifying.csv')

merged_df = pd.merge(races, pit_stops, on='raceId', how='inner')
df = pd.merge(merged_df, drivers, on='driverId', how='inner')
df = df.drop(columns=['url_x','url_y','number'])
df_new_merged = pd.merge(df, results, on=['driverId','raceId'], how='inner')
df_new_merged1 = pd.merge(df_new_merged, driver_standings, on=['driverId','raceId'], how='inner')
df_new_merged = pd.merge(df_new_merged, circuits, on=['circuitId'], how='inner')
condition1 = df_new_merged['year'] >= 2000
condition2 = df_new_merged['year'] <= 2023
df_new_merged1 = df_new_merged[condition1 & condition2 ]

df_new_merged1.to_csv('all.csv')
def convert_new_time_to_seconds_v4(time_str):
    if time_str == '\\N':
        return np.nan
    try:
        # Split by colon and reverse the list to start with seconds and go backwards to minutes, hours, etc.
        time_parts = list(map(float, time_str.split(':')))[::-1]
        total_seconds = sum(time_part * (60 ** index) for index, time_part in enumerate(time_parts))
        return total_seconds
    except ValueError:
        return np.nan

# Apply the function to the 'new_time' column
df_new_merged1['new_time_seconds'] = df_new_merged1['time'].replace('\+', '', regex=True).apply(convert_new_time_to
df_cleaned = df_new_merged1[['raceId','year', 'round','circuitId','circuit_type','circuit_type_num','driverId','gri
df_cleaned = df_cleaned.drop_duplicates()
df_cleaned.sort_values(by=['raceId', 'positionOrder'], inplace=True)
df_cleaned['cumulative_time_seconds'] = df_cleaned.groupby('raceId')['new_time_seconds'].cumsum()

# Creating a variable for each of the circuit types
df_cleaned['Race_Circuit'] = np.where(df_cleaned['circuit_type_num']== 3, 1, 0)
df_cleaned['Street_Circuit'] = np.where(df_cleaned['circuit_type_num']== 2, 1, 0)
```

```
df_cleaned['Road_Circuit'] = np.where(df_cleaned['circuit_type_num']== 1, 1, 0)
df_cleaned['rank'] = df_cleaned['rank'].astype(int)

#Dropping NULL values in Cumulative race finish times - Racers not able to finish race
df_cleaned = df_cleaned.fillna(0)
print(df_new_merged1.columns)
print(df_cleaned.columns)
df_cleaned.to_csv('clean.csv')
```

```
Index(['raceId', 'year', 'round', 'circuitId', 'name_x', 'date', 'time_x',
       'fp1_date', 'fp1_time', 'fp2_date', 'fp2_time', 'fp3_date', 'fp3_time',
       'quali_date', 'quali_time', 'sprint_date', 'sprint_time', 'driverId',
       'stop', 'lap', 'time_y', 'duration', 'milliseconds_x', 'driverRef',
       'code', 'forename', 'surname', 'dob', 'nationality', 'resultId',
       'constructorId', 'number', 'grid', 'position', 'positionText',
       'positionOrder', 'points', 'laps', 'time', 'milliseconds_y',
       'fastestLap', 'rank', 'fastestLapTime', 'fastestLapSpeed', 'statusId',
       'circuitRef', 'name_y', 'location', 'country', 'lat', 'lng', 'alt',
       'url', 'circuit_type', 'circuit_type_num', 'new_time_seconds'],
      dtype='object')
Index(['raceId', 'year', 'round', 'circuitId', 'circuit_type',
       'circuit_type_num', 'driverId', 'grid', 'rank', 'position',
       'positionOrder', 'time', 'new_time_seconds', 'stop', 'laps',
       'constructorId', 'points', 'cumulative_time_seconds', 'Race_Circuit',
       'Street_Circuit', 'Road_Circuit'],
      dtype='object')
```

```
/var/folders/mr/l3x9p9wd385gcyns864s9f2w0000gn/T/ipykernel_3325/1563000677.py:45: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  df_new_merged1['new_time_seconds'] = df_new_merged1['time'].replace('\+', '', regex=True).apply(convert_new_time_t
o_seconds_v4)
```

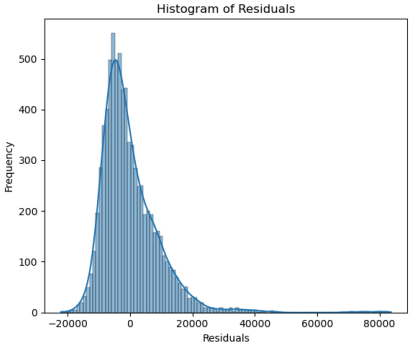
In [ ]:

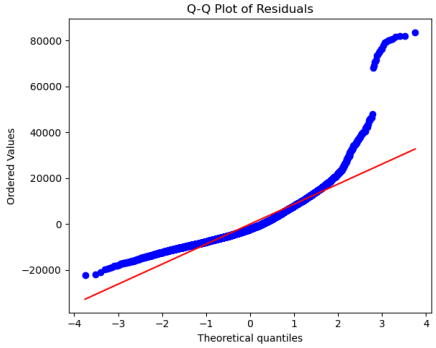
```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Assuming df_cleaned is your DataFrame
Xinitial = df_cleaned[['circuitId','driverId', 'grid', 'stop', 'rank', 'Race_Circuit', 'Street_Circuit', 'laps', 'y
yinitial = df_cleaned['cumulative_time_seconds']

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(Xinitial, yinitial, train_size=0.75, random_state=0)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)

# Fit the OLS model
model = sm.OLS(y_train, X_train)
results = model.fit()
print("OLS Model Summary:")
print(results.summary())

# To Handle multicolinearity
vif_data = pd.DataFrame()
vif_data["feature"] = X_train.columns
vif_data["VIF"] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
print(vif_data)

# Predicting on the training data
predicted_y = results.predict(X_train)
residuals = y_train - predicted_y

# Plotting the residuals
fig, ax = plt.subplots(1, 3, figsize=(18, 5))
# Histogram of residuals
sns.histplot(residuals, kde=True, ax=ax[0])
ax[0].set_title('Histogram of Residuals')
ax[0].set_xlabel('Residuals')
ax[0].set_ylabel('Frequency')
# Q-Q plot of residuals
stats.probplot(residuals, dist="norm", plot=ax[1])
ax[1].set_title('Q-Q Plot of Residuals')
# Residuals vs. fitted values
```

```
ax[2].scatter(results.fittedvalues, residuals)
ax[2].axhline(0, color='red', linestyle='dashed', linewidth=2)
ax[2].set_title('Residuals vs. Fitted Values')
ax[2].set_xlabel('Fitted Values')
ax[2].set_ylabel('Residuals')
plt.tight_layout()
plt.show()

# Perform the Breusch-Pagan Test
bp_test = het_breuschpagan(results.resid, results.model.exog)
measures = ('LM Statistic', 'LM-Test p-value', 'F-Statistic', 'F-Test p-value')
bp_results = dict(zip(measures, bp_test))
print("Breusch-Pagan Test Results:")
print(bp_results)
if bp_results['LM-Test p-value'] < 0.05 or bp_results['F-Test p-value'] < 0.05:
    print('The data has heteroskedasticity (p-value < 0.05). Applying Weighted Least Squares (WLS).')
    # Treat heteroskedasticity with WLS
    weights = 1 / (results.resid ** 2)
    wls_model = sm.WLS(y_train, X_train, weights=weights)
    wls_results = wls_model.fit()
    print("WLS Model Summary:")
    print(wls_results.summary())
else:
    print('The data does not have significant heteroskedasticity (p-value >= 0.05). No need to apply WLS.')

# Fit the OLS model with robust standard errors (regardless of heteroskedasticity result for comparison)
results_robust = results.get_robustcov_results(cov_type='HC3')
print("OLS Model with Robust Standard Errors Summary:")
print(results_robust.summary())
```

```
OLS Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:     cumulative_time_seconds   R-squared:                       0.220
Model:                                 OLS   Adj. R-squared:                  0.219
Method:                      Least Squares   F-statistic:                     202.2
Date:                     Thu, 01 Aug 2024   Prob (F-statistic):               0.00
Time:                             19:32:51   Log-Likelihood:                -83644.
No. Observations:                     7909   AIC:                         1.673e+05
Df Residuals:                         7897   BIC:                         1.674e+05
Df Model:                               11
Covariance Type:                 nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const         -4.2e+05   6.51e+04     -6.455      0.000   -5.48e+05   -2.92e+05
circuitId      -4.9844      4.890     -1.019      0.308     -14.570       4.601
driverId       -1.0574      0.323     -3.270      0.001      -1.691      -0.424
grid         -106.8572     23.664     -4.516      0.000    -153.244     -60.470
stop         2808.3748    113.637     24.713      0.000    2585.615    3031.134
rank         -220.0211     25.548     -8.612      0.000    -270.102    -169.941
Race_Circuit 4204.1824    481.257      8.736      0.000    3260.792    5147.572
Street_Circuit 3013.6038  493.085      6.112      0.000    2047.026    3980.182
laps          112.4768     10.010     11.236      0.000      92.854     132.100
year          206.4765     32.305      6.392      0.000     143.151     269.802
constructorId  -3.5041      1.342     -2.611      0.009      -6.135      -0.874
points        263.2711     21.316     12.351      0.000     221.486     305.056
==============================================================================
Omnibus:                     4383.835   Durbin-Watson:                   2.008
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            59525.424
Skew:                           2.372   Prob(JB):                         0.00
Kurtosis:                      15.575   Cond. No.                     1.28e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.28e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
          feature            VIF
0           const  372204.923657
1       circuitId       1.151255
2        driverId       1.382766
3            grid       1.944453
4            stop       1.021608
5            rank       1.991964
6    Race_Circuit       4.659900
7  Street_Circuit       4.465109
8            laps       1.108966
9            year       1.419734
10  constructorId       1.146245
11         points       2.113782
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```





```
Breusch-Pagan Test Results:
{'LM Statistic': 273.1788196782718, 'LM-Test p-value': 3.845312389470348e-52, 'F-Statistic': 25.683885656759657, 'F-
Test p-value': 4.224267665617925e-53}
The data has heteroskedasticity (p-value < 0.05). Applying Weighted Least Squares (WLS).
WLS Model Summary:
                            WLS Regression Results
==============================================================================
Dep. Variable:     cumulative_time_seconds   R-squared:                       0.999
Model:                                 WLS   Adj. R-squared:                  0.999
Method:                      Least Squares   F-statistic:                 6.676e+05
Date:                     Thu, 01 Aug 2024   Prob (F-statistic):               0.00
Time:                             19:32:52   Log-Likelihood:                -77721.
No. Observations:                     7909   AIC:                          1.555e+05
Df Residuals:                         7897   BIC:                          1.555e+05
Df Model:                               11
Covariance Type:                 nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -4.19e+05   3455.766   -121.248      0.000   -4.26e+05   -4.12e+05
circuitId     -4.7355      0.183    -25.853      0.000      -5.095      -4.376
driverId      -1.0509      0.014    -73.492      0.000      -1.079      -1.023
grid        -107.0254      1.225    -87.361      0.000    -109.427    -104.624
stop        2810.0779      6.841    410.797      0.000    2796.669    2823.487
rank        -220.5468      1.132   -194.787      0.000    -222.766    -218.327
Race_Circuit  4274.1317     30.006    142.441      0.000    4215.311    4332.952
Street_Circuit  3078.4677     31.167     98.773      0.000    3017.372    3139.563
laps         112.8767      0.394    286.160      0.000     112.104     113.650
year         205.9096      1.710    120.450      0.000     202.559     209.261
constructorId    -3.4535      0.050    -69.276      0.000      -3.551      -3.356
points       265.3157      1.066    248.853      0.000     263.226     267.406
==============================================================================
Omnibus:                    31156.029   Durbin-Watson:                   1.905
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1321.173
Skew:                           0.424   Prob(JB):                    1.29e-287
Kurtosis:                       1.186   Cond. No.                     4.10e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.1e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
OLS Model with Robust Standard Errors Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:     cumulative_time_seconds   R-squared:                       0.220
Model:                                 OLS   Adj. R-squared:                  0.219
Method:                      Least Squares   F-statistic:                     220.4
Date:                     Thu, 01 Aug 2024   Prob (F-statistic):               0.00
Time:                             19:32:52   Log-Likelihood:                -83644.
No. Observations:                     7909   AIC:                          1.673e+05
Df Residuals:                         7897   BIC:                          1.674e+05
Df Model:                               11
Covariance Type:                       HC3
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -4.2e+05   6.89e+04     -6.099      0.000   -5.55e+05   -2.85e+05
circuitId     -4.9844      4.541     -1.098      0.272     -13.885       3.916
driverId      -1.0574      0.309     -3.423      0.001      -1.663      -0.452
grid        -106.8572     26.367     -4.053      0.000    -158.543     -55.172
stop        2808.3748    157.227     17.862      0.000    2500.168    3116.582
rank        -220.0211     23.649     -9.303      0.000    -266.380    -173.662
Race_Circuit  4204.1824    378.752     11.100      0.000    3461.728    4946.637
Street_Circuit  3013.6038    360.517      8.359      0.000    2306.895    3720.313
laps         112.4768      9.931     11.325      0.000      93.009     131.945
year         206.4765     34.126      6.050      0.000     139.580     273.373
constructorId    -3.5041      1.306     -2.682      0.007      -6.065      -0.943
points       263.2711     21.935     12.002      0.000     220.272     306.270
==============================================================================
Omnibus:                    4383.835   Durbin-Watson:                   2.008
Prob(Omnibus):                 0.000   Jarque-Bera (JB):            59525.424
Skew:                          2.372   Prob(JB):                         0.00
Kurtosis:                     15.575   Cond. No.                     1.28e+06
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)
[2] The condition number is large, 1.28e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

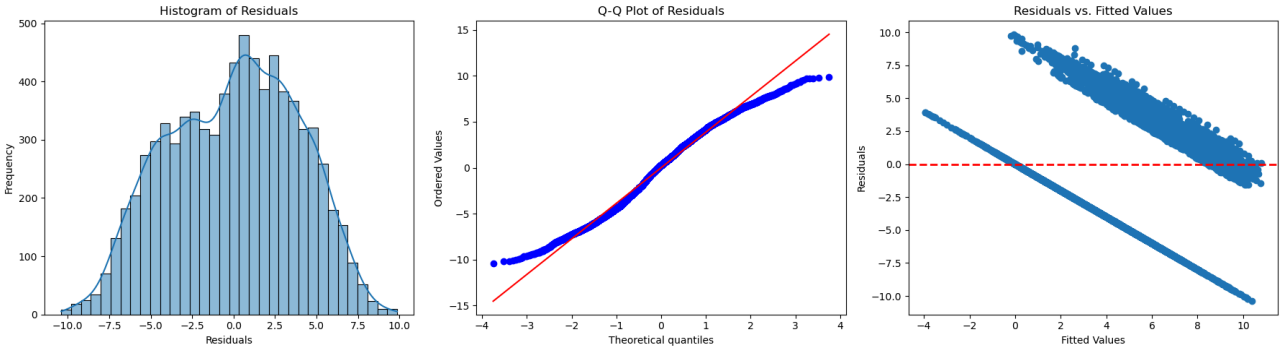
In [ ]:
```
# Hypothesis Testing for the different circuit: Road, Race and Street

hypothesis = '(Race_Circuit=0, Street_Circuit=0)'
#Pass the hypothesis to the Wald_Test
print(results_robust.wald_test(hypothesis))

#We do not embrace the null and there is a difference between the different types of track and the time it takes to
```

```
<F test: F=array([[62.26236896]]), p=1.4817501629469654e-27, df_denom=7.9e+03, df_num=2>
/opt/anaconda3/lib/python3.11/site-packages/statsmodels/base/model.py:1906: FutureWarning: The behavior of wald_test
will change after 0.14 to returning scalar test statistic values. To get the future behavior now, set scalar to Tru
e. To silence this message while retaining the legacy behavior, set scalar to False.
  warnings.warn(
```

In [ ]:
```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Assuming df_cleaned is your DataFrame

# Creating Dummy variables for splitting the two different types of circuits (street and race)
df_cleaned['Race_Circuit_grid'] = df_cleaned['Race_Circuit'] * df_cleaned['grid']
df_cleaned['Street_Circuit_grid'] = df_cleaned['Street_Circuit'] * df_cleaned['grid']



# Define the independent variables (X) and the dependent variable (y)
Xinitial = df_cleaned[['grid','stop', 'rank', 'Race_Circuit', 'Street_Circuit', 'laps', 'year', 'Race_Circuit_grid'
yinitial = df_cleaned['cumulative_time_seconds']

# Apply Natural Log Transformation to the dependent variable
yinitial_transformed = np.log(df_cleaned['cumulative_time_seconds'] + 1)
print(f"Skewness of transformed cumulative_time_seconds: {yinitial_transformed.skew()}")

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(Xinitial, yinitial_transformed, train_size=0.75, random_state=0

# Add a constant to the model (intercept)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)


# Calculate VIF for each predictor
vif_data = pd.DataFrame()
vif_data["feature"] = X_train.columns
vif_data["VIF"] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
print("VIF Data:")
print(vif_data)

# Fit the OLS model
model = sm.OLS(y_train, X_train)
results = model.fit()
print("OLS Model Summary with Transformed Dependent Variable:")
print(results.summary())

# Predicting on the training data
predicted_y = results.predict(X_train)
residuals = y_train - predicted_y

# Plotting the residuals
fig, ax = plt.subplots(1, 3, figsize=(18, 5))
# Histogram of residuals
sns.histplot(residuals, kde=True, ax=ax[0])
ax[0].set_title('Histogram of Residuals')
ax[0].set_xlabel('Residuals')
ax[0].set_ylabel('Frequency')
# Q-Q plot of residuals
stats.probplot(residuals, dist="norm", plot=ax[1])
ax[1].set_title('Q-Q Plot of Residuals')
# Residuals vs. fitted values
ax[2].scatter(results.fittedvalues, residuals)
ax[2].axhline(0, color='red', linestyle='dashed', linewidth=2)
ax[2].set_title('Residuals vs. Fitted Values')
ax[2].set_xlabel('Fitted Values')
ax[2].set_ylabel('Residuals')
```

```
plt.tight_layout()
plt.show()

# Perform the Breusch-Pagan Test
from statsmodels.stats.diagnostic import het_breuschpagan
bp_test = het_breuschpagan(results.resid, results.model.exog)
measures = ('LM Statistic', 'LM-Test p-value', 'F-Statistic', 'F-Test p-value')
bp_results = dict(zip(measures, bp_test))
print("Breusch-Pagan Test Results:")
print(bp_results)

# Apply Weighted Least Squares (WLS) if heteroskedasticity is detected
if bp_results['LM-Test p-value'] < 0.05 or bp_results['F-Test p-value'] < 0.05:
    print('The data has heteroskedasticity (p-value < 0.05). Applying Weighted Least Squares (WLS).')
    weights = 1 / (results.resid ** 2)
    wls_model = sm.WLS(y_train, X_train, weights=weights)
    wls_results = wls_model.fit()
    print("WLS Model Summary:")
    print(wls_results.summary())
    # Predicting on the test data using the WLS model
    y_pred_train_wls = wls_results.predict(X_train)
    y_pred_test_wls = wls_results.predict(X_test)
    # Perform RMSE and R2 test with Wald test on train data
    rmse_train_wls = np.sqrt(mean_squared_error(y_train, y_pred_train_wls))
    r2_train_wls = r2_score(y_train, y_pred_train_wls)
    # Perform RMSE and R2 test with Wald test on test data
    rmse_test_wls = np.sqrt(mean_squared_error(y_test, y_pred_test_wls))
    r2_test_wls = r2_score(y_test, y_pred_test_wls)

    # Output the WLS results
    print(f"Train RMSE (WLS): {rmse_train_wls}")
    print(f"Train R-squared (WLS): {r2_train_wls}")
    print(" ")
    print(f"Test RMSE (WLS): {rmse_test_wls}")
    print(f"Test R-squared (WLS): {r2_test_wls}")
else:
    print('The data does not have significant heteroskedasticity (p-value >= 0.05). No need to apply WLS.')
```

```
Skewness of transformed cumulative_time_seconds: -0.19598903775263365
VIF Data:
                feature            VIF
0                 const  270290.818699
1                  grid      18.445503
2                  stop       1.014493
3                  rank       1.593518
4         Race_Circuit      18.940374
5       Street_Circuit      18.810190
6                  laps       1.101929
7                  year       1.029900
8    Race_Circuit_grid      25.621225
9  Street_Circuit_grid      20.115156
OLS Model Summary with Transformed Dependent Variable:
                                OLS Regression Results
==============================================================================
Dep. Variable:     cumulative_time_seconds   R-squared:                       0.320
Model:                                 OLS   Adj. R-squared:                  0.320
Method:                      Least Squares   F-statistic:                     413.8
Date:                     Fri, 02 Aug 2024   Prob (F-statistic):               0.00
Time:                             11:54:55   Log-Likelihood:                -21975.
No. Observations:                     7909   AIC:                         4.397e+04
Df Residuals:                         7899   BIC:                         4.404e+04
Df Model:                                9
Covariance Type:                 nonrobust
==============================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                   -52.7075     22.782     -2.314      0.021     -97.367      -8.048
grid                     -0.3255      0.030    -10.872      0.000      -0.384      -0.267
stop                      0.1843      0.047      3.961      0.000       0.093       0.275
rank                     -0.2881      0.009    -30.695      0.000      -0.307      -0.270
Race_Circuit              0.0313      0.399      0.079      0.937      -0.750       0.813
Street_Circuit            0.0645      0.416      0.155      0.877      -0.751       0.880
laps                      0.0330      0.004      8.059      0.000       0.025       0.041
year                      0.0300      0.011      2.655      0.008       0.008       0.052
Race_Circuit_grid         0.1588      0.031      5.159      0.000       0.098       0.219
Street_Circuit_grid       0.1578      0.032      4.904      0.000       0.095       0.221
==============================================================================
Omnibus:                      620.985   Durbin-Watson:                   1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              217.506
Skew:                          -0.117   Prob(JB):                     5.88e-48
Kurtosis:                       2.222   Cond. No.                     1.05e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.05e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



```
Breusch-Pagan Test Results:
{'LM Statistic': 329.9577331592434, 'LM-Test p-value': 1.1355297659380292e-65, 'F-Statistic': 38.209696371501636, 'F
-Test p-value': 4.062596101603407e-67}
The data has heteroskedasticity (p-value < 0.05). Applying Weighted Least Squares (WLS).
WLS Model Summary:
                            WLS Regression Results
==============================================================================
Dep. Variable:     cumulative_time_seconds   R-squared:                       1.000
Model:                                 WLS   Adj. R-squared:                  1.000
Method:                      Least Squares   F-statistic:                 4.075e+06
Date:                     Fri, 02 Aug 2024   Prob (F-statistic):               0.00
Time:                             11:54:55   Log-Likelihood:                -17649.
No. Observations:                     7909   AIC:                             3.532e+04
Df Residuals:                         7899   BIC:                             3.539e+04
Df Model:                                9
Covariance Type:                 nonrobust
==============================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const                -53.3500      0.826    -64.562      0.000     -54.970     -51.730
grid                  -0.3269      0.001   -292.843      0.000      -0.329      -0.325
stop                   0.1854      0.001    125.829      0.000       0.182       0.188
rank                  -0.2880      0.000   -605.487      0.000      -0.289      -0.287
Race_Circuit           0.0343      0.003     11.248      0.000       0.028       0.040
Street_Circuit         0.0689      0.006     12.164      0.000       0.058       0.080
laps                   0.0331      0.000    268.035      0.000       0.033       0.033
year                   0.0303      0.000     74.172      0.000       0.030       0.031
Race_Circuit_grid      0.1601      0.001    151.774      0.000       0.158       0.162
Street_Circuit_grid    0.1586      0.001    138.726      0.000       0.156       0.161
==============================================================================
Omnibus:                    27486.376   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1315.616
Skew:                          -0.102   Prob(JB):                    2.08e-286
Kurtosis:                       1.012   Cond. No.                     3.86e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.86e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
Train RMSE (WLS): 3.8946488009693003
Train R-squared (WLS): 0.320412471941246

Test RMSE (WLS): 3.934348436267168
Test R-squared (WLS): 0.30854996208669827
```