In [737…

```
# Importing libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
import xgboost as xgb
```

In [738…

```
# Reading challenge 1 train data
df = pd.read_csv("Challenge1_train_data.csv")
```

In [491…

```
df.head(20)
```

Out[491…

	date	dc_name	size_code	retail_price	total_tires	zip_code
0	2022-02-05	OAKLAND	1856015	70.090	1	94604
1	2021-03-02	OAKLAND	1856015	59.090	1	94604
2	2020-12-08	OAKLAND	1856015	53.545	1	94604
3	2022-01-17	OAKLAND	1856015	58.410	1	94604
4	2020-11-13	OAKLAND	1856015	43.085	1	94604
5	2020-10-07	OAKLAND	1856015	76.910	1	94604
6	2020-12-14	OAKLAND	1856015	70.455	1	94604
7	2022-02-22	OAKLAND	1856015	84.590	1	94604
8	2022-04-18	OAKLAND	1856015	71.910	1	94604
9	2020-10-20	OAKLAND	1856015	67.455	1	94604
10	2021-08-31	OAKLAND	1856015	84.455	1	94604
11	2022-02-08	OAKLAND	1856015	72.785	1	94604
12	2021-12-16	OAKLAND	1856015	77.710	1	94604
13	2022-02-15	OAKLAND	1856015	95.910	1	94604
14	2022-06-14	OAKLAND	1856015	149.090	1	94604
15	2021-09-30	OAKLAND	1856015	78.410	1	94604
16	2022-01-10	OAKLAND	1856015	14.590	1	94604
17	2021-02-08	OAKLAND	1856015	96.910	1	94604
18	2022-06-09	OAKLAND	1856015	64.545	1	94604
19	2022-06-06	OAKLAND	1856015	103.910	1	94604

In [492…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284700 entries, 0 to 284699
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   date          284700 non-null  object
 1   dc_name       284700 non-null  object
 2   size_code     284700 non-null  int64
 3   retail_price  284666 non-null  float64
 4   total_tires   284700 non-null  int64
 5   zip_code      284700 non-null  int64
dtypes: float64(1), int64(3), object(2)
memory usage: 13.0+ MB
```

In [493…

```
df.shape
```

Out[493…

```
(284700, 6)
```

In [494…

```
# Data Explorations
# Explore "dc_name" columns
df["dc_name"].value_counts()
```

Out[494…

```
SACRAMENTO     113150
BAKERSFIELD     76650
OAKLAND         59860
SAN JOSE        35040
Name: dc_name, dtype: int64
```

In [495…

```
# Explore "size_code"
df["size_code"].value_counts()
```

Out[495…

```
2657516    2920
2257516    2920
2257515    2920
2354518    2920
2355018    2920
           ...
2454017     730
2557517     730
2653522     730
2158516     730
1955515     730
Name: size_code, Length: 157, dtype: int64
```

In [496…

```
# Explore "size_code"
df["zip_code"].unique()
```

Out[496…

```
array([94604, 95131, 95838, 93308], dtype=int64)
```

In [654…

```
# drop zip codes
#df.drop(["zip_code"], axis=1, inplace = True)
```

In [739…

```
# Convert date columns to Year, Month and Day
df.date = pd.to_datetime(df.date)
```

In [740…

```
# The model will not accept datetime, hence create a feature for each date part
df["Year"] = df["date"].dt.year
df["Month"] = df["date"].dt.month
df["Day"] = df["date"].dt.day
df["Day_of_week"] = df['date'].dt.day_name()
```

In [499…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284700 entries, 0 to 284699
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   date          284700 non-null  datetime64[ns]
 1   dc_name       284700 non-null  object        
 2   size_code     284700 non-null  int64         
 3   retail_price  284666 non-null  float64       
 4   total_tires   284700 non-null  int64         
 5   zip_code      284700 non-null  int64         
 6   Year          284700 non-null  int64         
 7   Month         284700 non-null  int64         
 8   Day           284700 non-null  int64         
 9   Day_of_week   284700 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(6), object(2)
memory usage: 21.7+ MB
```

In [500…

```python
from pandas.tseries.holiday import USFederalHolidayCalendar
```

In [741…

```python
cal = USFederalHolidayCalendar()
holidays = cal.holidays(start=df['date'].min(),
                        end=df['date'].max()).to_pydatetime()
df['holiday'] = df['date'].isin(holidays)
```

In [502…

```python
df
```

Out[502…

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_we
0	2022-02-05	OAKLAND	1856015	70.090	1	94604	2022	2	5	Satur
1	2021-03-02	OAKLAND	1856015	59.090	1	94604	2021	3	2	Tues
2	2020-12-08	OAKLAND	1856015	53.545	1	94604	2020	12	8	Tues
3	2022-01-17	OAKLAND	1856015	58.410	1	94604	2022	1	17	Mon
4	2020-11-13	OAKLAND	1856015	43.085	1	94604	2020	11	13	Fri
...	...	...	...	...	...	...	...	...	...	...
284695	2022-07-24	BAKERSFIELD	3512520	0.000	0	93308	2022	7	24	Sun
284696	2020-11-22	BAKERSFIELD	3512520	0.000	0	93308	2020	11	22	Sun
284697	2021-08-08	BAKERSFIELD	3512520	0.000	0	93308	2021	8	8	Sun
284698	2022-02-14	BAKERSFIELD	3512520	0.000	0	93308	2022	2	14	Mon
284699	2020-11-15	BAKERSFIELD	3512520	0.000	0	93308	2020	11	15	Sun

284700 rows × 11 columns

In [33]:

```python
# df[['year','month','day']] = df.date.apply(lambda x: pd.Series(x.strftime("%Y,%m,%d")
```

In [503...

```
df.head()
```

Out[503...

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_week	holi
0	2022-02-05	OAKLAND	1856015	70.090	1	94604	2022	2	5	Saturday	F
1	2021-03-02	OAKLAND	1856015	59.090	1	94604	2021	3	2	Tuesday	F
2	2020-12-08	OAKLAND	1856015	53.545	1	94604	2020	12	8	Tuesday	F
3	2022-01-17	OAKLAND	1856015	58.410	1	94604	2022	1	17	Monday	-
4	2020-11-13	OAKLAND	1856015	43.085	1	94604	2020	11	13	Friday	F

In [742...

```
df["season"] = df["Month"]%12 // 3 + 1
```

In [505...

```
df.head()
```

Out[505...

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_week	holi
0	2022-02-05	OAKLAND	1856015	70.090	1	94604	2022	2	5	Saturday	F
1	2021-03-02	OAKLAND	1856015	59.090	1	94604	2021	3	2	Tuesday	F
2	2020-12-08	OAKLAND	1856015	53.545	1	94604	2020	12	8	Tuesday	F
3	2022-01-17	OAKLAND	1856015	58.410	1	94604	2022	1	17	Monday	-
4	2020-11-13	OAKLAND	1856015	43.085	1	94604	2020	11	13	Friday	F

In [660...

```
#df["season"] = df["season"].astype(str)
```

In [661...

```
#cols_to_transform = ["season","Day_of_week","holiday","dc_name"]
```

In [662...

```
#df  = pd.get_dummies( df, columns = cols_to_transform )
```

In [663...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284700 entries, 0 to 284699
Data columns (total 24 columns):
 #   Column                    Non-Null Count   Dtype         
---  ------                    --------------   -----         
 0   date                      284700 non-null  datetime64[ns]
 1   size_code                 284700 non-null  int64         
 2   retail_price              284666 non-null  float64       
 3   total_tires           284700 non-null  int64
 4   Year                  284700 non-null  int64
 5   Month                 284700 non-null  int64
 6   Day                   284700 non-null  int64
 7   season_1              284700 non-null  uint8
 8   season_2              284700 non-null  uint8
 9   season_3              284700 non-null  uint8
 10  season_4              284700 non-null  uint8
 11  Day_of_week_Friday    284700 non-null  uint8
 12  Day_of_week_Monday    284700 non-null  uint8
 13  Day_of_week_Saturday  284700 non-null  uint8
 14  Day_of_week_Sunday    284700 non-null  uint8
 15  Day_of_week_Thursday  284700 non-null  uint8
 16  Day_of_week_Tuesday   284700 non-null  uint8
 17  Day_of_week_Wednesday 284700 non-null  uint8
 18  holiday_False         284700 non-null  uint8
 19  holiday_True          284700 non-null  uint8
 20  dc_name_BAKERSFIELD   284700 non-null  uint8
 21  dc_name_OAKLAND       284700 non-null  uint8
 22  dc_name_SACRAMENTO    284700 non-null  uint8
 23  dc_name_SAN JOSE      284700 non-null  uint8
dtypes: datetime64[ns](1), float64(1), int64(5), uint8(17)
memory usage: 19.8 MB
```

In [664…

```
#df = df.reindex(columns = [col for col in df.columns if col != 'total_tires'] + ['tota
```

In [743…

```
# change object data type to category
# Represent dc_name as numbers to avoid text values
df["dc_name_cat"] = pd.Categorical(df["dc_name"])
df["dc_name_num"] = df["dc_name_cat"].cat.codes
```

In [744…

```
df["Day_of_week_cat"] = pd.Categorical(df["Day_of_week"])
df["Day_of_week_num"] = df["Day_of_week_cat"].cat.codes
```

In [745…

```
df["holiday_cat"] = pd.Categorical(df["holiday"])
df["holiday_num"] = df["holiday_cat"].cat.codes
```

In [691…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284700 entries, 0 to 284699
Data columns (total 18 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   date             284700 non-null  datetime64[ns]
 1   dc_name          284700 non-null  object
 2   size_code        284700 non-null  int64
 3   retail_price     284666 non-null  float64
 4   total_tires      284700 non-null  int64
 5   zip_code         284700 non-null  int64
 6   Year             284700 non-null  int64
 7   Month            284700 non-null  int64
 8   Day              284700 non-null  int64
 9   Day_of_week      284700 non-null  object
 10  holiday          284700 non-null  bool
 11  season           284700 non-null  int64
 12  dc_name_cat      284700 non-null  category
 13  dc_name_num      284700 non-null  int8
 14  Day_of_week_cat  284700 non-null  category
 15  Day_of_week_num  284700 non-null  int8
 16  holiday_cat      284700 non-null  category
 17  holiday_num      284700 non-null  int8
```

```
dtypes: bool(1), category(3), datetime64[ns](1), float64(1), int64(7), int8(3), object
(2)
memory usage: 25.8+ MB
```

In [511…

```
# saving the dataframe
df.to_csv('clean_data.csv')
```

In [50]:

```
import xgboost as xgb
```

In [41]:

```
from xgboost import XGBRegressor
```

In [196…

```
# define model

model = XGBRegressor()
```

In [692…

```
df.columns
```

Out[692…

```
Index(['date', 'dc_name', 'size_code', 'retail_price', 'total_tires',
       'zip_code', 'Year', 'Month', 'Day', 'Day_of_week', 'holiday', 'season',
       'dc_name_cat', 'dc_name_num', 'Day_of_week_cat', 'Day_of_week_num',
       'holiday_cat', 'holiday_num'],
      dtype='object')
```

In [746…

```
train_split = 0.9
# Set the date at which to split train and eval data
# Of the unique dates available, pick the split between train and eval dates
dates_avail = df["date"].unique()
split_date_index = int(dates_avail.shape[0] * train_split)
split_date = dates_avail[split_date_index]
# Train data is on or before the split date
train_df = df.query("date > @split_date")
# And eval data is after
eval_df = df.query("date <= @split_date")

features = ['dc_name_num', 'size_code', 'retail_price',
     'Year', 'Month', 'Day', 'season',
          'Day_of_week_num',
        'holiday_num']
label = ["total_tires"]
x_train = train_df[features]
y_train = train_df[label]
x_eval = eval_df[features]
y_eval = eval_df[label]
```

In [707…

```
x_train.head()
```

Out[707…

	dc_name_num	size_code	retail_price	Year	Month	Day	season	Day_of_week_num	holiday_num
0	1	1856015	70.090	2022	2	5	1	2	0
1	1	1856015	59.090	2021	3	2	2	5	0
2	1	1856015	53.545	2020	12	8	1	5	0
3	1	1856015	58.410	2022	1	17	1	1	1
4	1	1856015	43.085	2020	11	13	4	0	0

In [708…

```
x_train.shape
```

Out[708…

```
(281580, 9)
```

In [709…

```
x_eval.shape
```

Out[709…

```
(3120, 9)
```

In [747…

```
# Build a model
model = xgb.XGBRegressor(
    n_estimators = 2000,
    max_depth = 25,
    min_child_weight = 10,
    learning_rate = 0.1
)
```

In [748…

```
# fit the model
model.fit(
    x_train,
    y_train,
    eval_set = [(x_train, y_train), (x_eval, y_eval)],
    early_stopping_rounds = 20,
    verbose = False
)
```

Out[748…

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.1, max_delta_step=0,
             max_depth=25, min_child_weight=10, missing=nan,
             monotone_constraints='()', n_estimators=2000, n_jobs=4,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

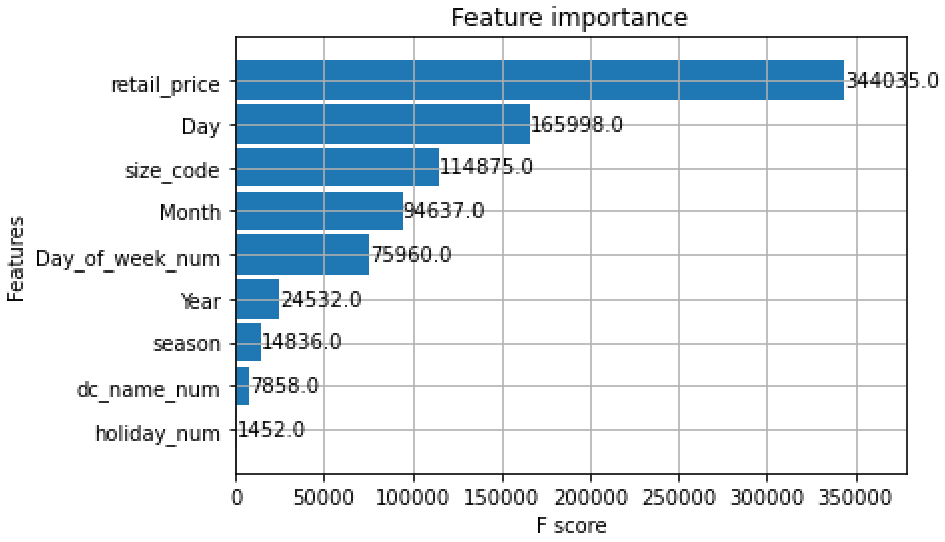
In [ ]:

In [749…

```
# Check feature importance
xgb.plot_importance(model, height=0.9)
```

Out[749…

```
<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>
```



Model Evaluation

In [750...

```
# Create a final dataframe to verify the predictions
df_pred = x_eval.copy()
# Recreate a column for the complete date
date_columns = ["Year", "Month", "Day"]
df_pred["Date"] = pd.to_datetime(df_pred[date_columns])
df_pred.drop(date_columns, inplace=True, axis=1)
```

In [519...

```
x_eval.head()
```

Out[519...

	Year	Month	Day	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price
112	2020	9	24	1	4	0	4	1856015	71.410
208	2020	9	23	1	6	0	4	1856015	45.090
292	2020	9	22	1	5	0	4	1856015	57.910
307	2020	9	25	1	0	0	4	1856015	67.455
384	2020	9	26	1	2	0	4	1856015	77.580

In [520...

```
df_pred.head()
```

Out[520...

	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price	Date
112	1	4	0	4	1856015	71.410	2020-09-24
208	1	6	0	4	1856015	45.090	2020-09-23
292	1	5	0	4	1856015	57.910	2020-09-22
307	1	0	0	4	1856015	67.455	2020-09-25
384	1	2	0	4	1856015	77.580	2020-09-26

In [751...

```
# Predict data for the eval dataset and save the predicted total_tires as a new column
df_pred["total_tires_Pred"] = model.predict(x_eval)
```

In [752...

```
df_pred["total_tires_Pred"] = round(df_pred["total_tires_Pred"])
```

In [753...

```
x_train["total_tires_Pred"] = model.predict(x_train)
```

```
<ipython-input-753-088a14b4799e>:1: 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#returning-a-view-versus-a-copy
  x_train["total_tires_Pred"] = model.predict(x_train)
```

In [482...

```
x_train.head()
```

Out[482...

	Year	Month	Day	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price
0	2022	2	5	1	2	0	1	1856015	70.090

	Year	Month	Day	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price
1	2021	3	2	1	5	0	2	1856015	59.090
2	2020	12	8	1	5	0	1	1856015	53.545
3	2022	1	17	1	1	1	1	1856015	58.410
4	2020	11	13	1	0	0	4	1856015	43.085

In [645…

```
y_train.head()
```

Out[645…

```
0    1
1    1
2    1
3    1
4    1
Name: total_tires, dtype: int64
```

In [646…

```
y_train.shape
```

Out[646…

```
(281580,)
```

In [325…

```
df_pred.head()
```

Out[325…

	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price	Date	total_tires_P
112	1	4	0	4	1856015	71.410	2020-09-24	2.514
208	1	6	0	4	1856015	45.090	2020-09-23	4.766
292	1	5	0	4	1856015	57.910	2020-09-22	-1.098
307	1	0	0	4	1856015	67.455	2020-09-25	-0.893
384	1	2	0	4	1856015	77.580	2020-09-26	5.675

In [754…

```
# Add the true Adj Close into the dataset
# Indexes were not reset, so we can join on the index (the indexes are reset during the
df_pred = df_pred.merge(
    y_eval,
    how = "inner",
    left_index = True,
    right_index = True
)
```

In [596…

```
df_pred.head()
```

Out[596…

	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price	Date	total_tires_P
112	1	4	0	4	1856015	71.410	2020-09-24	

	dc_name_num	Day_of_week_num	holiday_num	season	size_code	retail_price	Date	total_tires_P
208	1	6	0	4	1856015	45.090	2020-09-23	
292	1	5	0	4	1856015	57.910	2020-09-22	
307	1	0	0	4	1856015	67.455	2020-09-25	
384	1	2	0	4	1856015	77.580	2020-09-26	

In [755…

```
# Calculate wape
def simple_wape(y_true, y_pred):
    """Calculates simple wape"""
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    return np.round(
        abs(y_true - y_pred).sum() / abs(y_true).sum()
        if abs(y_true).sum() != 0
        else np.inf,
        5,
    )
```

In [756…

```
simple_wape(df_pred["total_tires"],df_pred["total_tires_Pred"])
```

Out[756…

```
0.32364
```

In [757…

```
simple_wape(y_train["total_tires"],x_train["total_tires_Pred"])
```

Out[757…

```
0.15495
```

Hyperparameters tunning

In [96]:

```
#! pip install hyperopt
```

```
Collecting hyperopt
[notice] A new release of pip available: 22.2.2 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
  Downloading hyperopt-0.2.7-py2.py3-none-any.whl (1.6 MB)
     ---------------------------------------- 1.6/1.6 MB 4.4 MB/s eta 0:00:00
Requirement already satisfied: networkx>=2.2 in c:\users\shobh\anaconda3\lib\site-packag
es (from hyperopt) (2.5)
Collecting py4j
  Downloading py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
     --------------------------------------- 200.5/200.5 kB 4.0 MB/s eta 0:00:00
Requirement already satisfied: numpy in c:\users\shobh\anaconda3\lib\site-packages (from
hyperopt) (1.22.3)
Requirement already satisfied: future in c:\users\shobh\anaconda3\lib\site-packages (fro
m hyperopt) (0.18.2)
Requirement already satisfied: scipy in c:\users\shobh\anaconda3\lib\site-packages (from
hyperopt) (1.5.2)
Requirement already satisfied: six in c:\users\shobh\anaconda3\lib\site-packages (from h
yperopt) (1.15.0)
Requirement already satisfied: cloudpickle in c:\users\shobh\anaconda3\lib\site-packages
```

```
(from hyperopt) (1.6.0)
Requirement already satisfied: tqdm in c:\users\shobh\anaconda3\lib\site-packages (from
hyperopt) (4.50.2)
Requirement already satisfied: decorator>=4.3.0 in c:\users\shobh\anaconda3\lib\site-pac
kages (from networkx>=2.2->hyperopt) (4.4.2)
Installing collected packages: py4j, hyperopt
Successfully installed hyperopt-0.2.7 py4j-0.10.9.7
```

In [97]:

```python
# import packages for hyperparameters tuning
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
```

In [99]:

```python
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10, 1),
        'n_estimators': 180,
        'seed': 0
    }
```

In [107...

```python
def objective(space):
    fc_model=xgb.XGBRegressor(
                    n_estimators =space['n_estimators'], max_depth = int(space['max_dep
                    reg_alpha = int(space['reg_alpha']),min_child_weight=int(space['min
                    colsample_bytree=int(space['colsample_bytree']))

    evaluation = [(x_train, y_train), (x_eval, y_eval)]

    fc_model.fit(x_train, y_train,
            eval_set=evaluation, eval_metric='auc',
            early_stopping_rounds=10,verbose=False)


    pred = fc_model.predict(x_eval)

    accuracy = simple_wape(y_eval, pred)
    print ("SCORE:", accuracy)
    return {'loss': -accuracy, 'status': STATUS_OK }
```

In [123...

```python
from sklearn.pipeline import Pipeline
pipe = Pipeline(steps=[
                       ("model", xgb.XGBRegressor(objective= 'reg:squarederror',
                                                  learning_rate = 0.1,
                                                  n_estimators =400,
                                                  max_depth = 3,
                                                  seed = 0))])
```

In [449...

```python
from sklearn.model_selection import RandomizedSearchCV
hyperparameter_grid = {
    'model__n_estimators': [100, 400, 800],
    'model__max_depth': [3, 6, 9],
    'model__learning_rate': [0.05, 0.1, 0.20],
    }

pipeline = RandomizedSearchCV(
    Pipeline(steps=[
                ("model", xgb.XGBRegressor(objective= 'reg:squarederror',seed = 0))
                ]),
    param_distributions=hyperparameter_grid,
    n_iter=20,
    scoring='r2',
    n_jobs=-1,
    cv=3,
    verbose=3)
```

In [ ]:
```
# submission
```

In [338…
```
df_forcast = df[df.date.between("2022-09-19", "2022-09-26")]
```

In [339…
```
df_forcast.head()
```

Out[339…

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_week
68	2022-09-19	OAKLAND	1856015	50.545	2	94604	2022	9	19	Monday
1058	2022-09-19	OAKLAND	1856515	95.340	16	94604	2022	9	19	Monday
1585	2022-09-19	OAKLAND	1956015	74.740	15	94604	2022	9	19	Monday
1914	2022-09-19	OAKLAND	1956515	68.650	16	94604	2022	9	19	Monday
2536	2022-09-19	OAKLAND	2055017	86.455	4	94604	2022	9	19	Monday

In [345…
```
df.head()
```

Out[345…

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_week	holi
0	2022-02-05	OAKLAND	1856015	70.090	1	94604	2022	2	5	Saturday	F
1	2021-03-02	OAKLAND	1856015	59.090	1	94604	2021	3	2	Tuesday	F
2	2020-12-08	OAKLAND	1856015	53.545	1	94604	2020	12	8	Tuesday	F
3	2022-01-17	OAKLAND	1856015	58.410	1	94604	2022	1	17	Monday	T
4	2020-11-13	OAKLAND	1856015	43.085	1	94604	2020	11	13	Friday	F

In [375…
```
df[(df["dc_name"]=="OAKLAND")]["size_code"].unique()
```

Out[375…
```
82
```

In [362…
```
df1  = pd.DataFrame({'size_code':[1856015, 1856515, 1956015, 1956515, 2055017, 2055516,
        2056515, 2056516, 2057015, 2057016, 2057514, 2057515, 2154517,
        2155017, 2155516, 2155517, 2155518, 2156016, 2156017, 2156516,
        2156517, 2157015, 2157016, 2254018, 2254517, 2254518, 2255017,
        2255018, 2255517, 2255518, 2255519, 2256016, 2256017, 2256018,
        2256516, 2256517, 2257515, 2257516, 2354019, 2354518, 2355017,
        2355018, 2355517, 2355518, 2355519, 2355520, 2356017, 2356018,
        2356516, 2356517, 2356518, 2357016, 2357515, 2358016, 2358516,
        2454019, 2454518, 2454519, 2454520, 2455020, 2456018, 2456517,
        2457017, 2457516, 2457517, 2555020, 2555520, 2556019, 2556518,
        2655020, 2656018, 2656518, 2657016, 2657017, 2657516, 2755520,
        2756020, 2756518, 2757018, 2854522, 2857017]})
```

In [363...

```
df1["dc_name"]="OAKLAND"
```

In [364...

```
df1.head()
```

Out[364...

	size_code	dc_name
0	1856015	OAKLAND
1	1856515	OAKLAND
2	1956015	OAKLAND
3	1956515	OAKLAND
4	2055017	OAKLAND

In [367...

```
df2 = pd.DataFrame({'date':['2022-09-20','2022-09-21','2022-09-22','2022-09-23','2022-0
```

In [369...

```
# Now to perform cross join, we will create
# a key column in both the DataFrames to
# merge on that key.
df1['key'] = 1
df2['key'] = 1

# to obtain the cross join we will merge on
# the key and drop it.
df3 = pd.merge(df1, df2, on ='key').drop("key", 1)
```

In [371...

```
df3.head(10)
```

Out[371...

	size_code	dc_name	date
0	1856015	OAKLAND	2022-09-20
1	1856015	OAKLAND	2022-09-21
2	1856015	OAKLAND	2022-09-22
3	1856015	OAKLAND	2022-09-23
4	1856015	OAKLAND	2022-09-24
5	1856015	OAKLAND	2022-09-25
6	1856015	OAKLAND	2022-09-26
7	1856515	OAKLAND	2022-09-20
8	1856515	OAKLAND	2022-09-21
9	1856515	OAKLAND	2022-09-22

In [374…

```
df3.shape
```

Out[374…

```
(574, 3)
```

In [379…

```
# Location 'SACRAMENTO'
df[(df["dc_name"]=="SACRAMENTO")]["size_code"].unique()
```

Out[379…

```
array([   11225,  1756514,  1756515,  1757014,  1855515,  1855516,
        1856015,  1856514,  1856515,  1857014,  1955515,  1955516,
        1956015,  1956515,  1957014,  2054517,  2055016,  2055017,
        2055516,  2055517,  2056015,  2056016,  2056515,  2056516,
        2057015,  2057016,  2057514,  2057515,  2154018,  2154517,
        2154518,  2155017,  2155516,  2155517,  2155518,  2156016,
        2156017,  2156516,  2156517,  2157015,  2157016,  2157515,
        2158516,  2254018,  2254019,  2254517,  2254518,  2254519,
        2255017,  2255018,  2255516,  2255517,  2255518,  2255519,
        2256016,  2256017,  2256018,  2256516,  2256517,  2257015,
        2257016,  2257515,  2257516,  2353519,  2354018,  2354019,
        2354517,  2354518,  2354519,  2355017,  2355018,  2355019,
        2355517,  2355518,  2355519,  2355520,  2356016,  2356017,
        2356018,  2356516,  2356517,  2356518,  2357016,  2357515,
        2358016,  2358017,  2358516,  2453519,  2453520,  2454017,
        2454018,  2454019,  2454020,  2454517,  2454518,  2454519,
        2454520,  2455018,  2455020,  2455519,  2456018,  2456517,
        2457016,  2457017,  2457516,  2457517,  2553518,  2553519,
        2553520,  2554018,  2554019,  2554020,  2554519,  2554520,
        2555019,  2555020,  2555518,  2555520,  2556517,  2556518,
        2557016,  2557018,  2653518,  2653519,  2653522,  2654520,
        2655020,  2656018,  2656517,  2656518,  2657016,  2657017,
        2657018,  2657516,  2753020,  2753519,  2753520,  2754019,
        2754020,  2754520,  2755520,  2756020,  2756518,  2756520,
        2757018,  2854522,  2857017,  2857516,  3054022,  3110515,
        3153520,  3512517,  3512518,  3512520, 22570195], dtype=int64)
```

In [380…

```
df4  = pd.DataFrame({'size_code':[   11225,  1756514,  1756515,  1757014,  1855515,  18
        1856015,  1856514,  1856515,  1857014,  1955515,  1955516,
        1956015,  1956515,  1957014,  2054517,  2055016,  2055017,
        2055516,  2055517,  2056015,  2056016,  2056515,  2056516,
        2057015,  2057016,  2057514,  2057515,  2154018,  2154517,
        2154518,  2155017,  2155516,  2155517,  2155518,  2156016,
        2156017,  2156516,  2156517,  2157015,  2157016,  2157515,
        2158516,  2254018,  2254019,  2254517,  2254518,  2254519,
        2255017,  2255018,  2255516,  2255517,  2255518,  2255519,
        2256016,  2256017,  2256018,  2256516,  2256517,  2257015,
        2257016,  2257515,  2257516,  2353519,  2354018,  2354019,
        2354517,  2354518,  2354519,  2355017,  2355018,  2355019,
        2355517,  2355518,  2355519,  2355520,  2356016,  2356017,
        2356018,  2356516,  2356517,  2356518,  2357016,  2357515,
        2358016,  2358017,  2358516,  2453519,  2453520,  2454017,
        2454018,  2454019,  2454020,  2454517,  2454518,  2454519,
        2454520,  2455018,  2455020,  2455519,  2456018,  2456517,
        2457016,  2457017,  2457516,  2457517,  2553518,  2553519,
        2553520,  2554018,  2554019,  2554020,  2554519,  2554520,
        2555019,  2555020,  2555518,  2555520,  2556517,  2556518,
        2557016,  2557018,  2653518,  2653519,  2653522,  2654520,
        2655020,  2656018,  2656517,  2656518,  2657016,  2657017,
        2657018,  2657516,  2753020,  2753519,  2753520,  2754019,
        2754020,  2754520,  2755520,  2756020,  2756518,  2756520,
        2757018,  2854522,  2857017,  2857516,  3054022,  3110515,
        3153520,  3512517,  3512518,  3512520, 22570195]})
```

In [382…

```
df4["dc_name"]="SACRAMENTO"
```

In [383…

```
# Now to perform cross join, we will create
df4['key'] = 1
df2['key'] = 1

# to obtain the cross join we will merge on
# the key and drop it.
df5 = pd.merge(df4, df2, on ='key').drop("key", 1)
```

In [384…

```
df5.head()
```

Out[384…

	size_code	dc_name	date
0	11225	SACRAMENTO	2022-09-20
1	11225	SACRAMENTO	2022-09-21
2	11225	SACRAMENTO	2022-09-22
3	11225	SACRAMENTO	2022-09-23
4	11225	SACRAMENTO	2022-09-24

In [385…

```
df5.shape
```

Out[385…

```
(1085, 3)
```

In [390…

```
#location Bakersfield
df[(df["dc_name"]=="BAKERSFIELD")]["size_code"].unique()
```

Out[390…

```
array([1856015, 1856514, 1856515, 1956015, 1956515, 2055016, 2055017,
       2055516, 2056016, 2056515, 2056516, 2057015, 2057016, 2057514,
       2057515, 2154517, 2155017, 2155516, 2155517, 2155518, 2156016,
       2156017, 2156516, 2156517, 2157015, 2157016, 2254018, 2254517,
       2254518, 2255017, 2255018, 2255517, 2255518, 2255519, 2256016,
       2256017, 2256018, 2256516, 2256517, 2257016, 2257515, 2257516,
       2354018, 2354019, 2354517, 2354518, 2354519, 2355017, 2355018,
       2355019, 2355517, 2355518, 2355519, 2355520, 2356016, 2356017,
       2356018, 2356516, 2356517, 2356518, 2357016, 2357515, 2358016,
       2358017, 2358516, 2454018, 2454019, 2454020, 2454517, 2454518,
       2454519, 2454520, 2455020, 2455519, 2456018, 2456517, 2457016,
       2457017, 2457516, 2457517, 2554519, 2554520, 2555020, 2555520,
       2556019, 2556517, 2556518, 2557517, 2655020, 2656018, 2656518,
       2657016, 2657017, 2657018, 2657516, 2754020, 2755520, 2756020,
       2756518, 2756520, 2757018, 2854522, 2857017, 2857516, 3512520],
      dtype=int64)
```

In [391…

```
df6  = pd.DataFrame({'size_code':[1856015, 1856514, 1856515, 1956015, 1956515, 2055016,
       2055516, 2056016, 2056515, 2056516, 2057015, 2057016, 2057514,
       2057515, 2154517, 2155017, 2155516, 2155517, 2155518, 2156016,
       2156017, 2156516, 2156517, 2157015, 2157016, 2254018, 2254517,
       2254518, 2255017, 2255018, 2255517, 2255518, 2255519, 2256016,
       2256017, 2256018, 2256516, 2256517, 2257016, 2257515, 2257516,
       2354018, 2354019, 2354517, 2354518, 2354519, 2355017, 2355018,
       2355019, 2355517, 2355518, 2355519, 2355520, 2356016, 2356017,
       2356018, 2356516, 2356517, 2356518, 2357016, 2357515, 2358016,
       2358017, 2358516, 2454018, 2454019, 2454020, 2454517, 2454518,
       2454519, 2454520, 2455020, 2455519, 2456018, 2456517, 2457016,
       2457017, 2457516, 2457517, 2554519, 2554520, 2555020, 2555520,
       2556019, 2556517, 2556518, 2557517, 2655020, 2656018, 2656518,
       2657016, 2657017, 2657018, 2657516, 2754020, 2755520, 2756020,
       2756518, 2756520, 2757018, 2854522, 2857017, 2857516, 3512520]})
```

In [392…

```
df6["dc_name"]="BAKERSFIELD"
```

In [393…

```
# Now to perform cross join, we will create
df6['key'] = 1
df2['key'] = 1

# to obtain the cross join we will merge on
# the key and drop it.
df7 = pd.merge(df6, df2, on ='key').drop("key", 1)
```

In [394…

```
df7.head()
```

Out[394…

	size_code	dc_name	date
0	1856015	BAKERSFIELD	2022-09-20
1	1856015	BAKERSFIELD	2022-09-21
2	1856015	BAKERSFIELD	2022-09-22
3	1856015	BAKERSFIELD	2022-09-23
4	1856015	BAKERSFIELD	2022-09-24

In [395…

```
df7.shape
```

Out[395…

```
(735, 3)
```

In [396…

```
df["dc_name"].unique()
```

Out[396…

```
array(['OAKLAND', 'SAN JOSE', 'SACRAMENTO', 'BAKERSFIELD'], dtype=object)
```

In [401…

```
# Location 'SAN JOSE'
df[(df["dc_name"]=="SAN JOSE")]["size_code"].unique()
```

Out[401…

```
48
```

In [398…

```
df8  = pd.DataFrame({'size_code':[   11225,  1956515,  2055516,  2056016,  2056516,  20
         2057515,  2154517,  2155516,  2155517,  2156016,  2254018,
         2254517,  2254518,  2255017,  2255517,  2256017,  2256517,
         2257515,  2257516,  2354518,  2355018,  2356018,  2356517,
         2356518,  2358016,  2358017,  2358516,  2454518,  2454520,
         2456018,  2457017,  2457516,  2457517,  2555020,  2656018,
         2657016,  2657017,  2657516,  2755520,  2756020,  2756518,
         2756520,  2757018,  2854522,  2857017,  3512520, 22570195]})
```

In [399…

```
df8["dc_name"]="SAN JOSE"
```

In [400…

```
# Now to perform cross join, we will create
df8['key'] = 1
df2['key'] = 1

# to obtain the cross join we will merge on
# the key and drop it.
df9 = pd.merge(df8, df2, on ='key').drop("key", 1)
```

In [402…

```
df9.head()
```

Out[402…

	size_code	dc_name	date
0	11225	SAN JOSE	2022-09-20
1	11225	SAN JOSE	2022-09-21
2	11225	SAN JOSE	2022-09-22
3	11225	SAN JOSE	2022-09-23
4	11225	SAN JOSE	2022-09-24

In [403…

```
df9.shape
```

Out[403…

```
(336, 3)
```

In [404…

```
df_test = pd.concat([df3,df5,df7,df9])
```

In [405…

```
df_test.head()
```

Out[405…

	size_code	dc_name	date
0	1856015	OAKLAND	2022-09-20
1	1856015	OAKLAND	2022-09-21
2	1856015	OAKLAND	2022-09-22
3	1856015	OAKLAND	2022-09-23
4	1856015	OAKLAND	2022-09-24

In [406…

```
df_test.shape
```

Out[406…

```
(2730, 3)
```

In [407…

```
# Convert date columns to Year, Month and Day
df_test.date = pd.to_datetime(df_test.date)
```

In [354…

```
df_groupby.head()
```

Out[354…

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_we
0	2022-02-05	OAKLAND	1856015	70.090	1	94604	2022	2	5	Saturc
1	2021-03-02	OAKLAND	1856015	59.090	1	94604	2021	3	2	Tuesc
2	2020-12-08	OAKLAND	1856015	53.545	1	94604	2020	12	8	Tuesc
3	2022-01-17	OAKLAND	1856015	58.410	1	94604	2022	1	17	Monc

	date	dc_name	size_code	retail_price	total_tires	zip_code	Year	Month	Day	Day_of_we
4	2020-11-13	OAKLAND	1856015	43.085	1	94604	2020	11	13	Fri
...	...	...	...	...	...	...	...	...	...	
236130	2022-02-18	BAKERSFIELD	3512520	426.090	1	93308	2022	2	18	Fri
236131	2021-08-06	BAKERSFIELD	3512520	312.590	1	93308	2021	8	6	Fri
236132	2022-04-01	BAKERSFIELD	3512520	406.410	1	93308	2022	4	1	Fri
236133	2022-05-13	BAKERSFIELD	3512520	475.090	2	93308	2022	5	13	Fri
236134	2022-08-04	BAKERSFIELD	3512520	500.910	2	93308	2022	8	4	Thurs

1950 rows × 18 columns

In [ ]:

```
df_submission =
```

In [341...

```
df["size_code"].unique()
```

Out[341...

```
array([ 1856015,  1856515,  1956015,  1956515,  2055017,  2055516,
        2056016,  2056515,  2056516,  2057015,  2057016,  2057514,
        2057515,  2154517,  2155017,  2155516,  2155517,  2155518,
        2156016,  2156017,  2156516,  2156517,  2157015,  2157016,
        2254018,  2254517,  2254518,  2255017,  2255018,  2255517,
        2255518,  2255519,  2256016,  2256017,  2256018,  2256516,
        2256517,  2257515,  2257516,  2354019,  2354518,  2355017,
        2355018,  2355517,  2355518,  2355519,  2355520,  2356017,
        2356018,  2356516,  2356517,  2356518,  2357016,  2357515,
        2358016,  2358516,  2454019,  2454518,  2454519,  2454520,
        2455020,  2456018,  2456517,  2457017,  2457516,  2457517,
        2555020,  2555520,  2556019,  2556518,  2655020,  2656018,
        2656518,  2657016,  2657017,  2657516,  2755520,  2756020,
        2756518,  2757018,  2854522,  2857017,    11225,  2358017,
        2756520,  3512520, 22570195,  1756514,  1756515,  1757014,
        1855515,  1855516,  1856514,  1857014,  1955515,  1955516,
        1957014,  2054517,  2055016,  2055517,  2056015,  2154018,
        2154518,  2157515,  2158516,  2254019,  2254519,  2255516,
        2257015,  2257016,  2353519,  2354018,  2354517,  2354519,
        2355019,  2356016,  2453519,  2453520,  2454017,  2454018,
        2454020,  2454517,  2455018,  2455519,  2457016,  2553518,
        2553519,  2553520,  2554018,  2554019,  2554020,  2554519,
        2554520,  2555019,  2555518,  2556517,  2557016,  2557018,
        2653518,  2653519,  2653522,  2654520,  2656517,  2657018,
        2753020,  2753519,  2753520,  2754019,  2754020,  2754520,
        2857516,  3054022,  3110515,  3153520,  3512517,  3512518,
        2557517], dtype=int64)
```

In [423...

```
df10 = df[["size_code","dc_name","retail_price"]].groupby(["size_code","dc_name"]).mean
```

In [424...

```
df10
```

Out[424…

		retail_price
size_code	**dc_name**	
11225	**SACRAMENTO**	260.104514
	SAN JOSE	254.235548
1756514	**SACRAMENTO**	46.230671
1756515	**SACRAMENTO**	71.053185
1757014	**SACRAMENTO**	46.390700
...	**...**	...
3512520	**BAKERSFIELD**	286.140610
	SACRAMENTO	310.921726
	SAN JOSE	311.242514
22570195	**SACRAMENTO**	205.721001
	SAN JOSE	194.767664

390 rows × 1 columns

In [420…

```
df10.head(2)
```

Out[420…

		retail_price	total_tires	zip_code	Year	Month	Day	holiday
size_code	**dc_name**							
11225	**SACRAMENTO**	260.104514	4.838356	95838.0	2021.217808	6.526027	15.720548	0.027397
	SAN JOSE	254.235548	5.921918	95131.0	2021.217808	6.526027	15.720548	0.027397

In [720…

```
# inner join
df11 = pd.merge(df_test, df10, on=['size_code',"dc_name"], how='inner')
```

In [649…

```
df11.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2730 entries, 0 to 2729
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   size_code     2730 non-null   int64
 1   dc_name       2730 non-null   object
 2   date          2730 non-null   datetime64[ns]
 3   retail_price  2730 non-null   float64
dtypes: datetime64[ns](1), float64(1), int64(1), object(1)
memory usage: 106.6+ KB
```

In [721…

```
# The model will not accept datetime, hence create a feature for each date part
df11["Year"] = df11["date"].dt.year
df11["Month"] = df11["date"].dt.month
df11["Day"] = df11["date"].dt.day
df11["Day_of_week"] = df11['date'].dt.day_name()
```

In [722...

```
df11["season"] = df11["Month"]%12 // 3 + 1
```

In [723...

```
cal = USFederalHolidayCalendar()
holidays = cal.holidays(start=df11['date'].min(),
                        end=df11['date'].max()).to_pydatetime()
df11['holiday'] = df11['date'].isin(holidays)
```

In [724...

```
df11.head()
```

Out[724...

	size_code	dc_name	date	retail_price	Year	Month	Day	Day_of_week	season	holiday
0	1856015	OAKLAND	2022-09-20	56.891904	2022	9	20	Tuesday	4	False
1	1856015	OAKLAND	2022-09-21	56.891904	2022	9	21	Wednesday	4	False
2	1856015	OAKLAND	2022-09-22	56.891904	2022	9	22	Thursday	4	False
3	1856015	OAKLAND	2022-09-23	56.891904	2022	9	23	Friday	4	False
4	1856015	OAKLAND	2022-09-24	56.891904	2022	9	24	Saturday	4	False

In [679...

```
df11  = pd.get_dummies( df11, columns = cols_to_transform )
```

In [725...

```
# change object data type to category
# Represent dc_name as numbers to avoid text values
df11["dc_name_cat"] = pd.Categorical(df11["dc_name"])
df11["dc_name_num"] = df11["dc_name_cat"].cat.codes
```

In [726...

```
df11["Day_of_week_cat"] = pd.Categorical(df11["Day_of_week"])
df11["Day_of_week_num"] = df11["Day_of_week_cat"].cat.codes
```

In [727...

```
df11["holiday_cat"] = pd.Categorical(df11["holiday"])
df11["holiday_num"] = df11["holiday_cat"].cat.codes
```

In [728...

```
df11.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2730 entries, 0 to 2729
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   size_code        2730 non-null   int64
 1   dc_name          2730 non-null   object
 2   date             2730 non-null   datetime64[ns]
 3   retail_price     2730 non-null   float64
 4   Year             2730 non-null   int64
 5   Month            2730 non-null   int64
 6   Day              2730 non-null   int64
 7   Day_of_week      2730 non-null   object
 8   season           2730 non-null   int64
 9   holiday          2730 non-null   bool
 10  dc_name_cat      2730 non-null   category
 11  dc_name_num      2730 non-null   int8
 12  Day_of_week_cat  2730 non-null   category
 13  Day_of_week_num  2730 non-null   int8
 14  holiday_cat      2730 non-null   category
 15  holiday_num      2730 non-null   int8
dtypes: bool(1), category(3), datetime64[ns](1), float64(1), int64(5), int8(3), object
(2)
memory usage: 232.6+ KB
```

In [729…

```
df11_eval = df11[features]
```

In [758…

```
df11["total_tires"] = model.predict(df11_eval)
```

In [731…

```
df11.head()
```

Out[731…

	size_code	dc_name	date	retail_price	Year	Month	Day	Day_of_week	season	holiday	dc_name
0	1856015	OAKLAND	2022-09-20	56.891904	2022	9	20	Tuesday	4	False	OAKL
1	1856015	OAKLAND	2022-09-21	56.891904	2022	9	21	Wednesday	4	False	OAKL
2	1856015	OAKLAND	2022-09-22	56.891904	2022	9	22	Thursday	4	False	OAKL
3	1856015	OAKLAND	2022-09-23	56.891904	2022	9	23	Friday	4	False	OAKL
4	1856015	OAKLAND	2022-09-24	56.891904	2022	9	24	Saturday	4	False	OAKL

In [759…

```
df_submission = df11[["date","dc_name","size_code","total_tires"]]
```

In [733…

```
df_submission.head()
```

Out[733…

	date	dc_name	size_code	total_tires
0	2022-09-20	OAKLAND	1856015	3.044636
1	2022-09-21	OAKLAND	1856015	3.780949
2	2022-09-22	OAKLAND	1856015	7.762942
3	2022-09-23	OAKLAND	1856015	7.433838
4	2022-09-24	OAKLAND	1856015	5.023912

In [446…

```
df_submission.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2730 entries, 0 to 2729
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   date         2730 non-null   object 
 1   dc_name      2730 non-null   object 
 2   size_code    2730 non-null   int64  
 3   total_tires  2730 non-null   float32
dtypes: float32(1), int64(1), object(2)
memory usage: 96.0+ KB
```

In [760…

```
df_submission["date"] = df_submission["date"].astype(str)
```

```
<ipython-input-760-420be9dff09b>:1: 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#returning-a-view-versus-a-copy
  df_submission["date"] = df_submission["date"].astype(str)
```

In [593…

```
df_submission.shape
```

Out[593…

```
(2730, 4)
```

In [761…

```
import requests
url = "https://scoring-app-uuzeqpiufa-ue.a.run.app/forecast/validate"
payload = {
"team_key": "xmb73wH9",
"data": df_submission.to_dict(orient="records")
}
response = requests.post(url, json=payload)
print(response.status_code, response.content)
```

```
200 b'{"message":"Success"}'
```

In [763…

```
import requests
url = "https://scoring-app-uuzeqpiufa-ue.a.run.app/forecast/submit"
payload = {
"team_key": "xmb73wH9",
"data": df_submission.to_dict(orient="records")
}
response = requests.post(url, json=payload)
print(response.status_code, response.content)
```

```
201 b'{"message":"Success","challenge":"forecast","score":1.89549}'
```

The End!