⇨ Part Four: XGBOOST

```
%%capture
!pip install numpy
```

```
%%capture
!pip install pandas
```

```
%%capture
!pip install xgboost
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from zipfile import ZipFile
from urllib.request import urlretrieve
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import random as rd
```

```
files = ("http://www.evanmarie.com/content/files/dataframes/merged_data.zip")
urlretrieve(files, 'merged_data.zip')
with ZipFile('merged_data.zip') as zipped_data:
  zipped_data.extractall(path='merged_data')
```

⍰ A little more housekeeping

```
orig_df_avg = pd.read_csv('merged_data/df_avgs.csv')
orig_df_zeros = pd.read_csv('merged_data/df_zeros.csv')
```

```
# Create categorical columns for moon and wkdy
orig_df_avg = pd.get_dummies(orig_df_avg, columns=['moon', 'wkdy'])
orig_df_zeros = pd.get_dummies(orig_df_zeros, columns=['moon', 'wkdy'])
```

```
keeper_cols = ['yr', 'mo', 'dy_cat', 'hr_bpm', 'o2_sat', 'o2_sat_kpa',
               'moon_First Quarter', 'moon_Full Moon', 'moon_Last Quarter',
               'moon_New Moon', 'moon_Waning Crescent', 'moon_Waning Gibbous',
               'moon_Waxing Crescent', 'moon_Waxing Gibbous', 'wkdy_Friday',
               'wkdy_Monday', 'wkdy_Saturday', 'wkdy_Sunday', 'wkdy_Thursday',
               'wkdy_Tuesday', 'wkdy_Wednesday']
```

```
orig_df_avg['targets'] = orig_df_avg['ex_min'] * orig_df_avg['act_cal'] * orig_df_avg['
```

```
xgboost_df = orig_df_avg.copy()
inputs = xgboost_df[keeper_cols]
targets = xgboost_df['targets']
```

⇨ XGBOOST TIME!

```
xgboost_scaler = MinMaxScaler().fit(inputs)
inputs = xgboost_scaler.transform(inputs)
```

```
train_inputs, test_inputs, train_targets, test_targets = train_test_split(inputs, targe
```

```
xgb_model = XGBRegressor(objective="reg:squarederror", random_state=42).fit(train_input
```

```
preds = xgb_model.predict(train_inputs)
```

TEST MODEL PARAMETERS:

```
# My Plotting Color Lists

bright01 = ['DeepPink', 'Aqua', 'Blue', 'Crimson', 'DarkMagenta', 'Fuchsia']
wild = ['#970FF2', '#0597F2', '#49D907', '#EAF205', '#F24607']
pleasant = ['#F23847', '#8B41F2', '#049DD9', '#04BF7B', '#F2CB05']
distinguished = ['#0B2B40', '#30A5BF', '#185359', '#F2BE22', '#A6874E']
reds_blues = ['#A6033F', '#0540F2', '#056CF2', '#05AFF2', '#BF2604']
daring = ['#462666', '#A957CF', '#FFEF00', '#D1FF45', '#200066']
easter_eggs = ['#BF7DFA', '#7472D6', '#85B4EB', '#81D2D6', '#81F7BE']
nice_calm = ['#4C4A59', '#1B7F7A', '#0897B4', '#4CABA6', '#F2CDAC']
clowns = ['#BF0B3B', '#D50BD9', '#151EBF', '#29A632', '#F28B0C']
crayons = ['#F21B54', '#AA19FC', '#1E7FC6', '#03A64A', '#F2E205']
classy_bright = ['#5F49F2', '#5079F2', '#F2B807', '#F28907', '#F2220F']
tasteful_bright = ['#F194A7', '#18B3FC', '#09D309', '#F6CB09', '#F24606']
pretty_pastels = ['#E4FFD3', '#77E8DC', '#8682FF', '#E882C2', '#FFD4B2']
eighties = ['#FF1B87', '#014DE8', '#0EFF16', '#E8A302', '#FF0169']
primarily = ['#E89609', '#FF1307', '#FFF300', '#0FE808', '#20ABFF']
calm_blues = ['#D2A2F2', '#C1B3F2', '#99C8F2', '#94E1F2', '#91F2F2']

color_groups = [bright01, wild, pleasant, distinguished, reds_blues, daring,
                easter_eggs, nice_calm, clowns, crayons, classy_bright,
               tasteful_bright, pretty_pastels, eighties, primarily, calm_blues]

individual_colors = ['DeepPink', 'Aqua', 'Blue', 'Crimson', 'DarkMagenta', 'Fuchsia',
                     '#970FF2', '#0597F2', '#49D907', '#EAF205', '#F24607', '#F23847',
                     '#8B41F2', '#049DD9', '#04BF7B', '#F2CB05', '#0B2B40', '#30A5BF',
                     '#185359', '#F2BE22', '#A6874E', '#0B2B40', '#30A5BF', '#185359',
```

```
                    '#F2BE22', '#A6874E', '#A6033F', '#0540F2', '#056CF2', '#05AFF2',
                    '#BF2604', '#462666', '#A957CF', '#FFEF00', '#D1FF45', '#200066',
                    '#BF7DFA', '#7472D6', '#85B4EB', '#81D2D6', '#81F7BE', '#4C4A59',
                    '#1B7F7A', '#0897B4', '#4CABA6', '#F2CDAC', '#BF0B3B', '#D50BD9',
                    '#151EBF', '#29A632', '#F28B0C', '#F21B54', '#AA19FC', '#1E7FC6',
                    '#03A64A', '#F2E205']

blueish = ['#00FFFF', '#0000FF', '#00FFFF', '#00CED1', '#00BFFF', '#0000CD', '#48D1CC',
                   '#B0E0E6', '#4169E1', '#87CEEB']
reddish = ['#FF00FF', '#FF69B4', '#FF00FF', '#FF4500', '#FF0000']
greenish = ['#7FFFD4', '#7FFF00', '#ADFF2F', '#00FF00', '#32CD32', '#3CB371', '#00FA9A'
                   '#FFA500', '#FFFF00']
other = ['#9932CC', '#9400D3', '#FF1493', '#BA55D3', '      #9370D8', '#7B68EE', '#DDA0D
facecolors = ['#101010', '#202020 ', '#2f2f2f', '#3f3f3f', '#4f4f4f', '#5f5f5f', '#6f6f
```

```
def test_parameters(**params):
    model = XGBRegressor(objective="reg:squarederror", n_jobs=-1, random_state=42, **pa
    model.fit(train_inputs, train_targets)
    train_rmse = rmse(model.predict(train_inputs), train_targets)
    val_rmse = rmse(model.predict(test_inputs), test_targets)
    return train_rmse, val_rmse
```

```
def compare_number_estimators_rmse(iterations, estimator_increment):

  comparison_num_estimators = []
  estimator_count = 10
  train_rmse = 0
  val_rmse = 0

  for iteration in range(iterations):
      train_rmse, val_rmse = test_parameters(n_estimators = estimator_count)
      comparison_num_estimators.append([iteration, estimator_count, train_rmse, val_rms
      print(f"XGBoost on iteration number {iteration + 1} of {iterations} with {estimat
      print(f"received a combined training RMSE of {train_rmse: .2f}")
      print(f"and a combined validation RMSE of {val_rmse: .2f}. \n")
      estimator_count += estimator_increment

  comparison_num_estimators = pd.DataFrame(comparison_num_estimators, columns = ['itera
  return(comparison_num_estimators)
```

n_estimators ⇨ RMSE comparison training vs validation

```
num_estimator_rmse = compare_number_estimators_rmse(25, 20)
```

```
XGBoost on iteration number 1 of 25 with 10 estimators
received a combined training RMSE of  1125.88
and a combined validation RMSE of  1248.26.


XGBoost on iteration number 2 of 25 with 30 estimators
```

```
received a combined training RMSE of  633.32
and a combined validation RMSE of  890.65.

XGBoost on iteration number 3 of 25 with 50 estimators
received a combined training RMSE of  424.58
and a combined validation RMSE of  771.40.

XGBoost on iteration number 4 of 25 with 70 estimators
received a combined training RMSE of  305.71
and a combined validation RMSE of  708.85.

XGBoost on iteration number 5 of 25 with 90 estimators
received a combined training RMSE of  235.48
and a combined validation RMSE of  668.00.

XGBoost on iteration number 6 of 25 with 110 estimators
received a combined training RMSE of  182.98
and a combined validation RMSE of  649.11.

XGBoost on iteration number 7 of 25 with 130 estimators
received a combined training RMSE of  142.63
and a combined validation RMSE of  631.25.

XGBoost on iteration number 8 of 25 with 150 estimators
received a combined training RMSE of  117.33
and a combined validation RMSE of  622.82.

XGBoost on iteration number 9 of 25 with 170 estimators
received a combined training RMSE of  93.78
and a combined validation RMSE of  616.09.

XGBoost on iteration number 10 of 25 with 190 estimators
received a combined training RMSE of  80.23
and a combined validation RMSE of  612.28.

XGBoost on iteration number 11 of 25 with 210 estimators
received a combined training RMSE of  65.58
and a combined validation RMSE of  609.62.

XGBoost on iteration number 12 of 25 with 230 estimators
received a combined training RMSE of  52.84
and a combined validation RMSE of  606.83.
```

```
XGBoost on iteration number 13 of 25 with 250 estimators
received a combined training RMSE of  45.08
and a combined validation RMSE of  604.87.

XGBoost on iteration number 14 of 25 with 270 estimators
received a combined training RMSE of  36.13
and a combined validation RMSE of  603.87.

XGBoost on iteration number 15 of 25 with 290 estimators
received a combined training RMSE of  30.31
and a combined validation RMSE of  603.51.

XGBoost on iteration number 16 of 25 with 310 estimators
received a combined training RMSE of  26.15
and a combined validation RMSE of  603.29.

XGBoost on iteration number 17 of 25 with 330 estimators
received a combined training RMSE of  22.83
and a combined validation RMSE of  603.25.

XGBoost on iteration number 18 of 25 with 350 estimators
received a combined training RMSE of  19.47
and a combined validation RMSE of  603.26.

XGBoost on iteration number 19 of 25 with 370 estimators
received a combined training RMSE of  16.19
and a combined validation RMSE of  603.02.

XGBoost on iteration number 20 of 25 with 390 estimators
received a combined training RMSE of  13.94
and a combined validation RMSE of  602.98.

XGBoost on iteration number 21 of 25 with 410 estimators
received a combined training RMSE of  12.07
and a combined validation RMSE of  602.91.

XGBoost on iteration number 22 of 25 with 430 estimators
received a combined training RMSE of  10.49
and a combined validation RMSE of  602.98.

XGBoost on iteration number 23 of 25 with 450 estimators
received a combined training RMSE of  9.38
and a combined validation RMSE of  602.87.
```

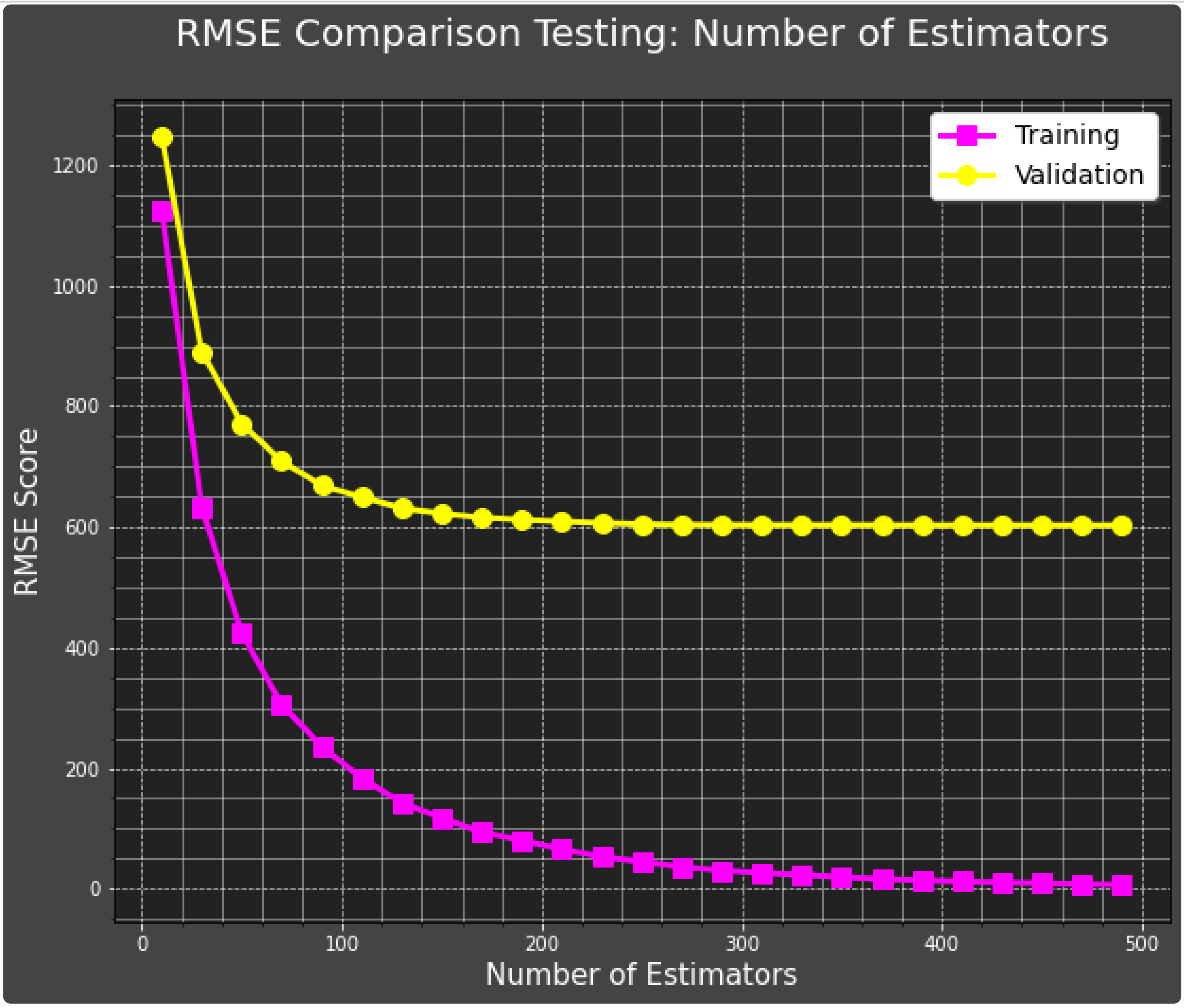
```
XGBoost on iteration number 24 of 25 with 470 estimators
received a combined training RMSE of  7.78
and a combined validation RMSE of  602.85.

XGBoost on iteration number 25 of 25 with 490 estimators
received a combined training RMSE of  6.80
and a combined validation RMSE of  602.84.
```

PLOT ONE PARAMETER FOR RMSE COMPARISON

plot_one_param_rmse(dataframe, key_param = None, plot_title=None, x_label=None, y_label=None, plot_y_lim = None, train_color = None, val_color = None, logscale_x = False)

```
def plot_one_param_rmse(dataframe, key_param = None, plot_title=None,
                        x_label=None, y_label=None, plot_y_lim = None,
                        train_color = None, val_color = None,
                        logscale_x = False):

  dataframe_to_training = dict(zip(dataframe[key_param], dataframe.training_rmse))
  dataframe_to_validation = dict(zip(dataframe[key_param], dataframe.validation_rmse))

  plt.figure(figsize=(10, 8), facecolor='#444444')
  ax = plt.axes()
  ax.set_facecolor("#222222")

  if logscale_x:
    plt.plot(np.log10(list(dataframe_to_training.keys())), list(dataframe_to_training.v
    plt.plot(np.log10(list(dataframe_to_validation.keys())), list(dataframe_to_validati

  else:
    plt.plot(list(dataframe_to_training.keys()), list(dataframe_to_training.values()),
    plt.plot(list(dataframe_to_validation.keys()), list(dataframe_to_validation.values(

  ax.minorticks_on()
  ax.grid(color='white', linestyle='--', linewidth=0.7)
  ax.tick_params(axis='x', colors='white')
  ax.tick_params(axis='y', colors='white')
  plt.ylim(plot_y_lim)
  plt.grid(which='minor', axis='both', alpha=0.5, color='white')
  plt.grid(which='major', axis='both', alpha=0.75, color='white')
  plt.rc('axes', titlesize=16)
  plt.xlabel(x_label, color='white', size=15)
  plt.ylabel(y_label, color='white', size=15)
  plt.title(plot_title, color="white", size = 20, y=1.05)
  ax.legend(shadow=True, fancybox=True, loc = 'upper right', fontsize=14)

  plt.show()
```

```
plot_one_param_rmse(num_estimator_rmse, key_param = 'num_estimators', plot_title="RMSE
                    x_label="Number of Estimators", y_label='RMSE Score',
                    train_color = rd.choice(reddish), val_color = rd.choice(greenish),
                    logscale_x = False)
```



n_estimators ⇨ Accuracy comparison training vs validation (max_depth is 44)

```
def num_estimators_accuracy(iterations, X_train, train_targets, X_val, val_targets, est

  comparison_num_estimators = []
  estimator_count = estimator_start
  training_accuracy = 0
  validation_accuracy = 0

  for iteration in range(iterations):
      model = XGBRegressor(objective="reg:squarederror", n_jobs=-1, random_state=42, n_
      model.fit(X_train, train_targets)
      comparison_num_estimators.append([iteration, estimator_count, model.score(X_train
      print("XGBoost Regressor Model with max depth of 44:")
      print(f"Iteration number {iteration + 1} of {iterations} with {estimator_count} c
      print(f"received a training score of {(model.score(X_train, train_targets)*100):
      estimator_count += estimator_increment
  comparison_num_estimators = pd.DataFrame(comparison_num_estimators, columns = ['itera
```

```
    return(comparison_num_estimators)
```

```
num_estimator_accuracy = num_estimators_accuracy(25, train_inputs, train_targets, test_
```

```
XGBoost Regressor Model with max depth of 44:
Iteration number 1 of 25 with 5 decision trees
received a training score of  90.75% and a validation score of  82.21%.

XGBoost Regressor Model with max depth of 44:
Iteration number 2 of 25 with 25 decision trees
received a training score of  100.00% and a validation score of  89.54%.

XGBoost Regressor Model with max depth of 44:
Iteration number 3 of 25 with 45 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 4 of 25 with 65 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 5 of 25 with 85 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 6 of 25 with 105 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 7 of 25 with 125 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 8 of 25 with 145 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 9 of 25 with 165 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 10 of 25 with 185 decision trees
received a training score of  100.00% and a validation score of  89.53%.
```

```
XGBoost Regressor Model with max depth of 44:
Iteration number 11 of 25 with 205 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 12 of 25 with 225 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 13 of 25 with 245 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 14 of 25 with 265 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 15 of 25 with 285 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 16 of 25 with 305 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 17 of 25 with 325 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 18 of 25 with 345 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 19 of 25 with 365 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 20 of 25 with 385 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 21 of 25 with 405 decision trees
```

```
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 22 of 25 with 425 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 23 of 25 with 445 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 24 of 25 with 465 decision trees
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with max depth of 44:
Iteration number 25 of 25 with 485 decision trees
received a training score of  100.00% and a validation score of  89.53%.
```

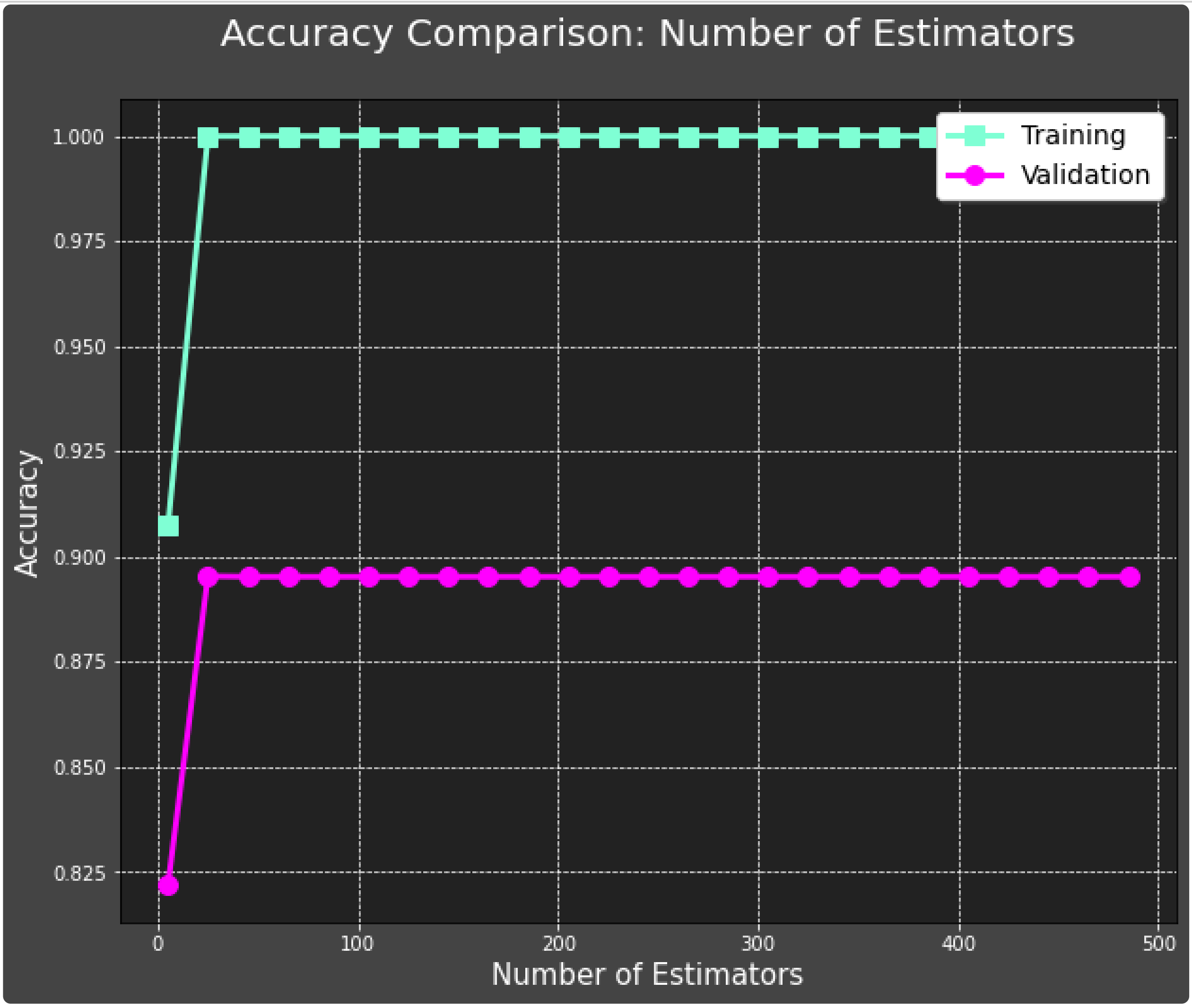
PLOT ONE PARAMETER FOR ACCURACY COMPARISON

`plot_one_param_accuracy(dataframe, key_param = None, plot_title=None, x_label=None, y_label=None, plot_y_lim = None, train_color = None, val_color = None, logscale_x = False)`

```
def plot_one_param_accuracy(dataframe, key_param = None, plot_title=None,
                  x_label=None, y_label=None, plot_y_lim = None,
                  train_color = None, val_color = None,
                  logscale_x = False):
    dataframe_to_training = dict(zip(dataframe[key_param], dataframe.training))
    dataframe_to_validation = dict(zip(dataframe[key_param], dataframe.validation))

    plt.figure(figsize=(10, 8), facecolor='#444444')
    ax = plt.axes()
    ax.set_facecolor("#222222")

    if logscale_x:
        plt.plot(np.log10(list(dataframe_to_training.keys())), list(dataframe_to_traini
        plt.plot(np.log10(list(dataframe_to_validation.keys())), list(dataframe_to_vali
    else:
        plt.plot(list(dataframe_to_training.keys()), list(dataframe_to_training.values(
        plt.plot(list(dataframe_to_validation.keys()), list(dataframe_to_validation.val

        ax.grid(color='white', linestyle='--', linewidth=0.7)
        ax.tick_params(axis='x', colors='white')
        ax.tick_params(axis='y', colors='white')
        plt.ylim(plot_y_lim)
        plt.rc('axes', titlesize=16)
```

```
        plt.xlabel(x_label, color='white', size=15)
        plt.ylabel(y_label, color='white', size=15)
        plt.title(plot_title, color="white", size = 20, y=1.05)
        ax.legend(shadow=True, fancybox=True, loc = 'upper right', fontsize=14)

    plt.show()
```

```
plot_one_param_accuracy(num_estimator_accuracy, key_param = "num_estimators", plot_titl
                  x_label="Number of Estimators", y_label="Accuracy", plot_y_lim = None
                  train_color = rd.choice(greenish), val_color = rd.choice(reddish),
                  logscale_x = False)
```



max_depth ⇨ RMSE comparison training vs validation (n_estimators is 200, based on its testing above.)

```
def compare_max_depth_rmse(iterations, start_depth, max_depth_increment):

  comparison_max_depth = []
  max_depth = start_depth
  train_rmse = 0
  val_rmse = 0

  for iteration in range(iterations):
      train_rmse, val_rmse = test_parameters(max_depth = max_depth, n_estimators = 200)
```

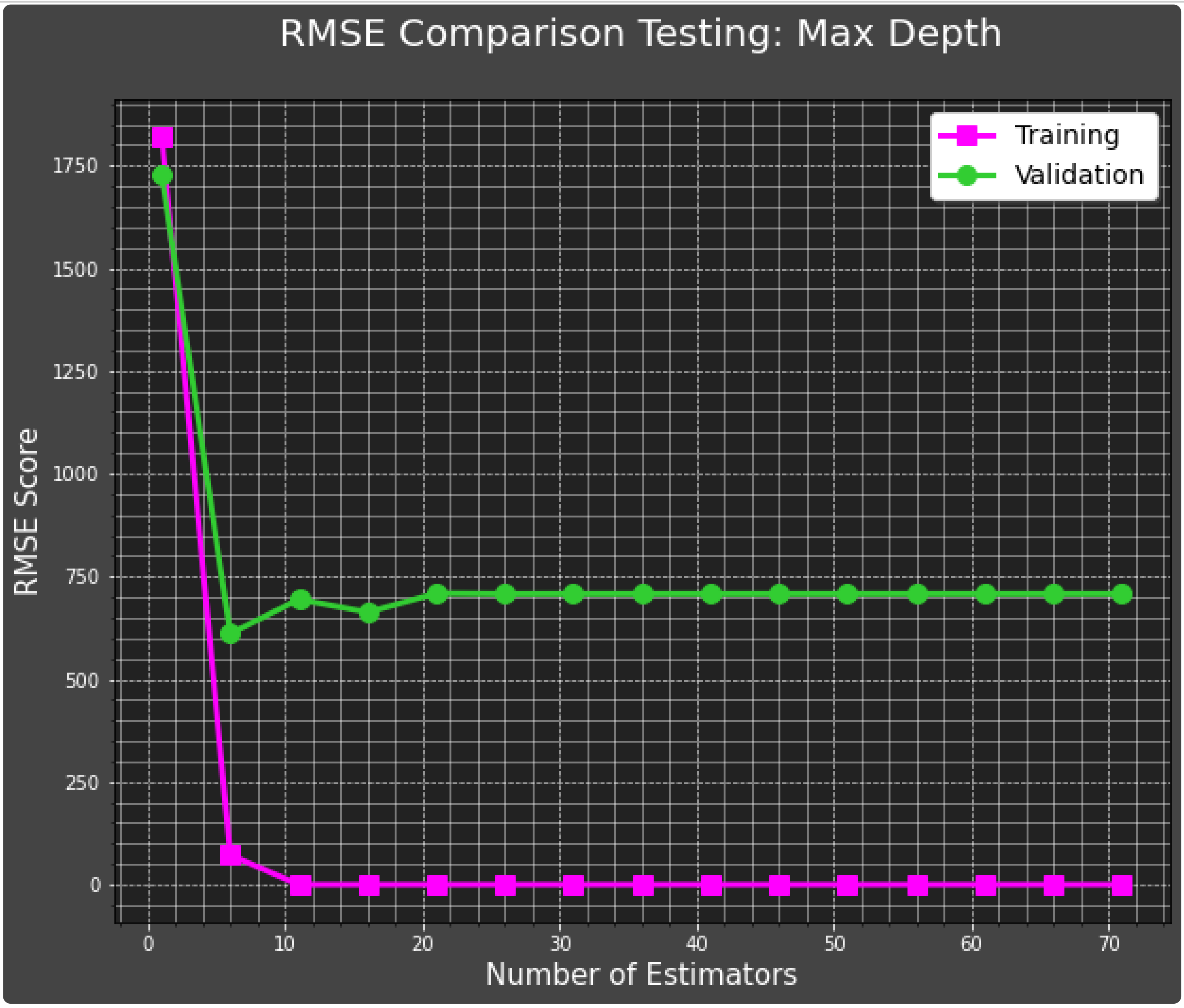
```
        comparison_max_depth.append([iteration, max_depth, train_rmse, val_rmse])
        print(f"XGBoost on iteration number {iteration + 1} of {iterations} with max dept
        print(f"received a training RMSE of {train_rmse: .2f} and a validation RMSE of {v

        max_depth += max_depth_increment

    comparison_max_depth = pd.DataFrame(comparison_max_depth, columns = ['iteration', 'ma
    return(comparison_max_depth)
```

```
max_depth_rmse = compare_max_depth_rmse(15, 1, 5)
```

```
XGBoost on iteration number 1 of 15 with max depth of 1
received a training RMSE of  1821.83 and a validation RMSE of  1727.99.

XGBoost on iteration number 2 of 15 with max depth of 6
received a training RMSE of  72.10 and a validation RMSE of  611.10.

XGBoost on iteration number 3 of 15 with max depth of 11
received a training RMSE of  0.01 and a validation RMSE of  694.58.

XGBoost on iteration number 4 of 15 with max depth of 16
received a training RMSE of  0.00 and a validation RMSE of  663.29.

XGBoost on iteration number 5 of 15 with max depth of 21
received a training RMSE of  0.00 and a validation RMSE of  709.25.

XGBoost on iteration number 6 of 15 with max depth of 26
received a training RMSE of  0.00 and a validation RMSE of  707.96.

XGBoost on iteration number 7 of 15 with max depth of 31
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 8 of 15 with max depth of 36
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 9 of 15 with max depth of 41
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 10 of 15 with max depth of 46
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 11 of 15 with max depth of 51
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 12 of 15 with max depth of 56
```

```
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 13 of 15 with max depth of 61
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 14 of 15 with max depth of 66
received a training RMSE of  0.00 and a validation RMSE of  708.16.

XGBoost on iteration number 15 of 15 with max depth of 71
received a training RMSE of  0.00 and a validation RMSE of  708.16.
```

```python
plot_one_param_rmse(max_depth_rmse, key_param = 'max_depth', plot_title="RMSE Compariso
                    x_label="Number of Estimators", y_label='RMSE Score',
                    train_color = rd.choice(reddish), val_color = rd.choice(greenish),
                    logscale_x = False)
```



max_depth ⇨ Accuracy comparison training vs validation (n_estimators is 200, based on its testing above.)

```python
def max_depth_accuracy(iterations, X_train, train_targets, X_val, val_targets, max_dept
```

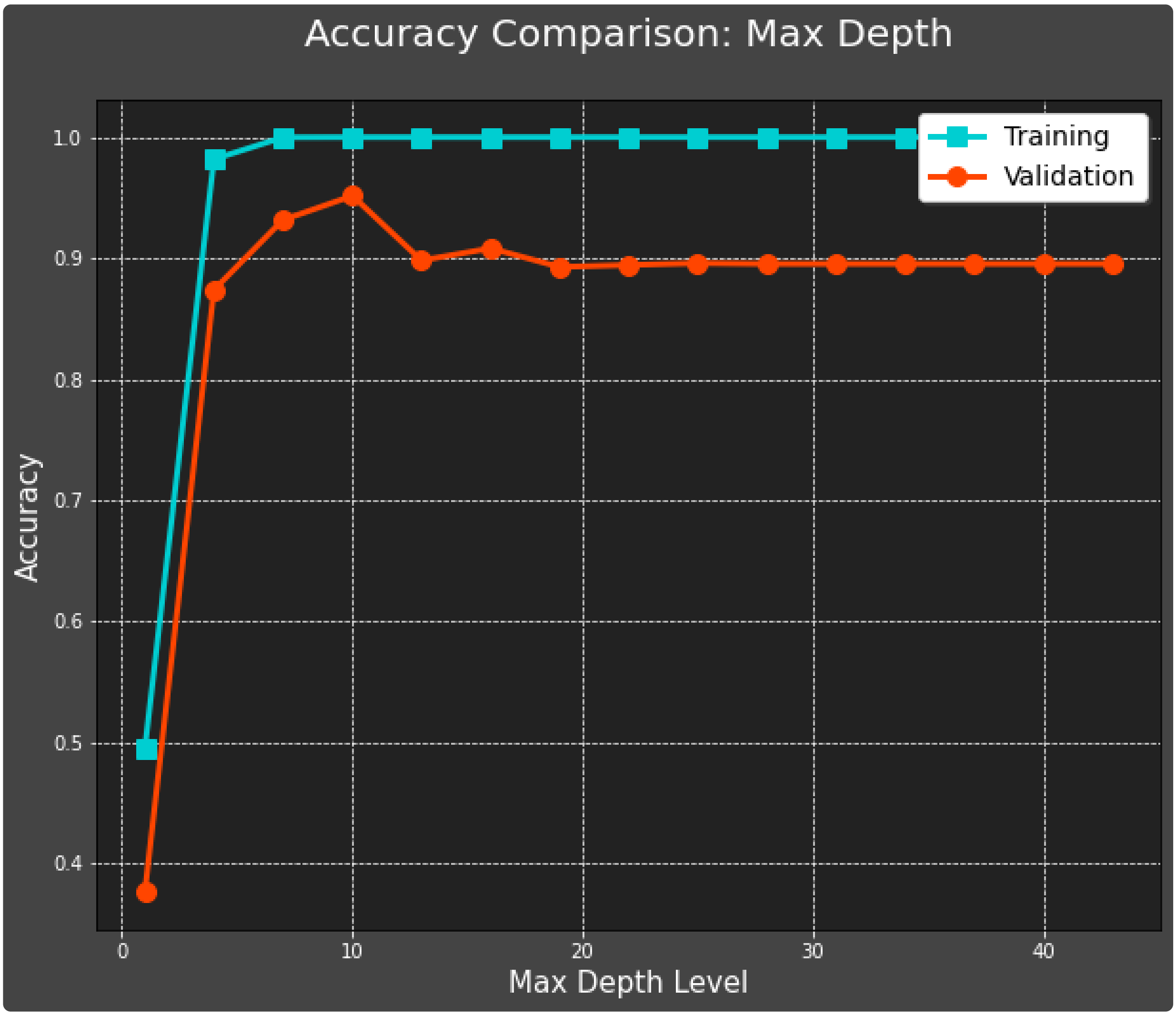
```
    comparison_max_depth = []
    max_depth = max_depth_start
    training_accuracy = 0
    validation_accuracy = 0

    for iteration in range(iterations):
        model = XGBRegressor(objective="reg:squarederror", n_jobs=-1, random_state=42, n_
        model.fit(X_train, train_targets)
        comparison_max_depth.append([iteration, max_depth, model.score(X_train, train_tar
        print("XGBoost Regressor Model with 200 decision trees:")
        print(f"Iteration number {iteration + 1} of {iterations} with max_depth of {max_d
        print(f"received a training score of {(model.score(X_train, train_targets)*100):
        max_depth += max_depth_increment
    comparison_max_depth = pd.DataFrame(comparison_max_depth, columns = ['iteration', 'ma

    return(comparison_max_depth)
```

```
max_depth_accuracy = max_depth_accuracy(15, train_inputs, train_targets, test_inputs, t
```

```
XGBoost Regressor Model with 200 decision trees:
Iteration number 1 of 15 with max_depth of 1:
received a training score of  49.54% and a validation score of  37.67%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 2 of 15 with max_depth of 4:
received a training score of  98.18% and a validation score of  87.38%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 3 of 15 with max_depth of 7:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 4 of 15 with max_depth of 10:
received a training score of  100.00% and a validation score of  95.15%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 5 of 15 with max_depth of 13:
received a training score of  100.00% and a validation score of  89.81%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 6 of 15 with max_depth of 16:
received a training score of  100.00% and a validation score of  90.82%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 7 of 15 with max_depth of 19:
received a training score of  100.00% and a validation score of  89.27%.
```

```
XGBoost Regressor Model with 200 decision trees:
Iteration number 8 of 15 with max_depth of 22:
received a training score of  100.00% and a validation score of  89.44%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 9 of 15 with max_depth of 25:
received a training score of  100.00% and a validation score of  89.59%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 10 of 15 with max_depth of 28:
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 11 of 15 with max_depth of 31:
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 12 of 15 with max_depth of 34:
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 13 of 15 with max_depth of 37:
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 14 of 15 with max_depth of 40:
received a training score of  100.00% and a validation score of  89.53%.

XGBoost Regressor Model with 200 decision trees:
Iteration number 15 of 15 with max_depth of 43:
received a training score of  100.00% and a validation score of  89.53%.
```

```
plot_one_param_accuracy(max_depth_accuracy, key_param = "max_depth", plot_title="Accura
                  x_label="Max Depth Level", y_label="Accuracy", plot_y_lim = None,
                  train_color = rd.choice(blueish), val_color = rd.choice(reddish),
                  logscale_x = False)
```



`learning_rate` ⇨ RMSE comparison training vs validation (n_estimators is 200, and max_depth is 7, based on the testing above.)

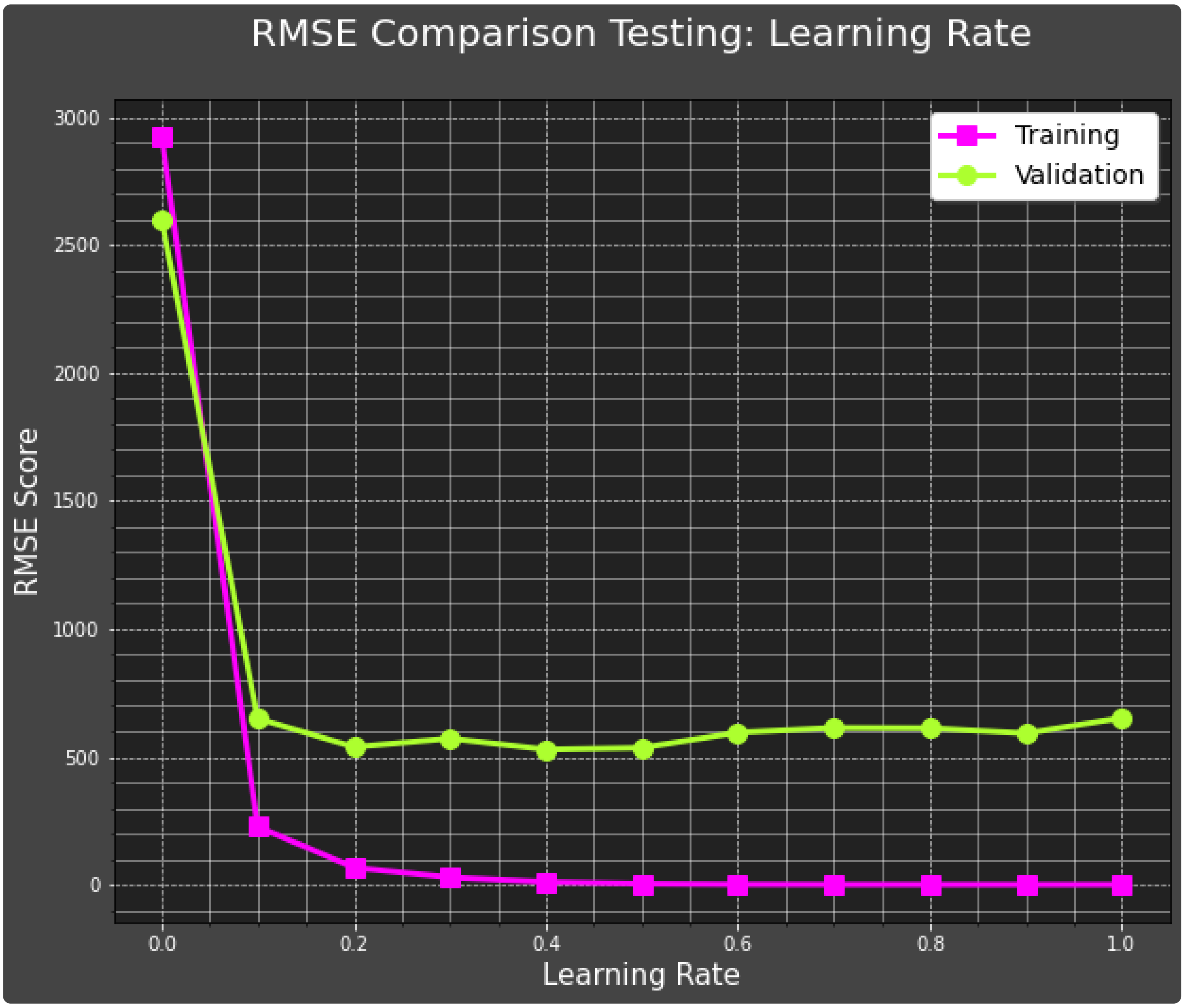
```
def compare_learning_rate_rmse(iterations, learning_rate_start, learning_rate_increment

  comparison_learning_rate = []
  learning_rate = learning_rate_start
  train_rmse = 0
  val_rmse = 0

  for iteration in range(iterations):
      train_rmse, val_rmse = test_parameters(learning_rate = learning_rate, n_estimator
      comparison_learning_rate.append([iteration, learning_rate, train_rmse, val_rmse])
      print(f"XGBoost on iteration number {iteration + 1} of {iterations} with learning
      print(f"received a training RMSE of {train_rmse: .2f} and a validation RMSE of {v
      learning_rate += learning_rate_increment

  comparison_learning_rate = pd.DataFrame(comparison_learning_rate, columns = ['iterati
  return(comparison_learning_rate)
```

```
learning_rate_rmse = compare_learning_rate_rmse(iterations = 11, learning_rate_start =
```

```
XGBoost on iteration number 1 of 11 with learning rate of 0.0
```

```
received a training RMSE of  2927.22 and a validation RMSE of  2600.15.

XGBoost on iteration number 2 of 11 with learning rate of 0.1
received a training RMSE of  227.03 and a validation RMSE of  650.09.

XGBoost on iteration number 3 of 11 with learning rate of 0.2
received a training RMSE of  66.46 and a validation RMSE of  538.30.

XGBoost on iteration number 4 of 11 with learning rate of 0.30000000000000004
received a training RMSE of  28.49 and a validation RMSE of  571.92.

XGBoost on iteration number 5 of 11 with learning rate of 0.4
received a training RMSE of  10.45 and a validation RMSE of  528.71.

XGBoost on iteration number 6 of 11 with learning rate of 0.5
received a training RMSE of  3.59 and a validation RMSE of  535.28.

XGBoost on iteration number 7 of 11 with learning rate of 0.6
received a training RMSE of  1.16 and a validation RMSE of  595.16.

XGBoost on iteration number 8 of 11 with learning rate of 0.7
received a training RMSE of  0.36 and a validation RMSE of  613.63.

XGBoost on iteration number 9 of 11 with learning rate of 0.7999999999999999
received a training RMSE of  0.21 and a validation RMSE of  612.49.

XGBoost on iteration number 10 of 11 with learning rate of 0.8999999999999999
received a training RMSE of  0.05 and a validation RMSE of  592.13.

XGBoost on iteration number 11 of 11 with learning rate of 0.9999999999999999
received a training RMSE of  0.01 and a validation RMSE of  652.26.
```

```
plot_one_param_rmse(learning_rate_rmse, key_param = 'learning_rate', plot_title="RMSE C
                   x_label="Learning Rate", y_label='RMSE Score',
                   train_color = rd.choice(reddish), val_color = rd.choice(greenish),
                   logscale_x = False)
```



`learning_rate` ⇨ Accuracy comparison training vs validation (n_estimators is 200, and max_depth is 7, based on the testing above.)

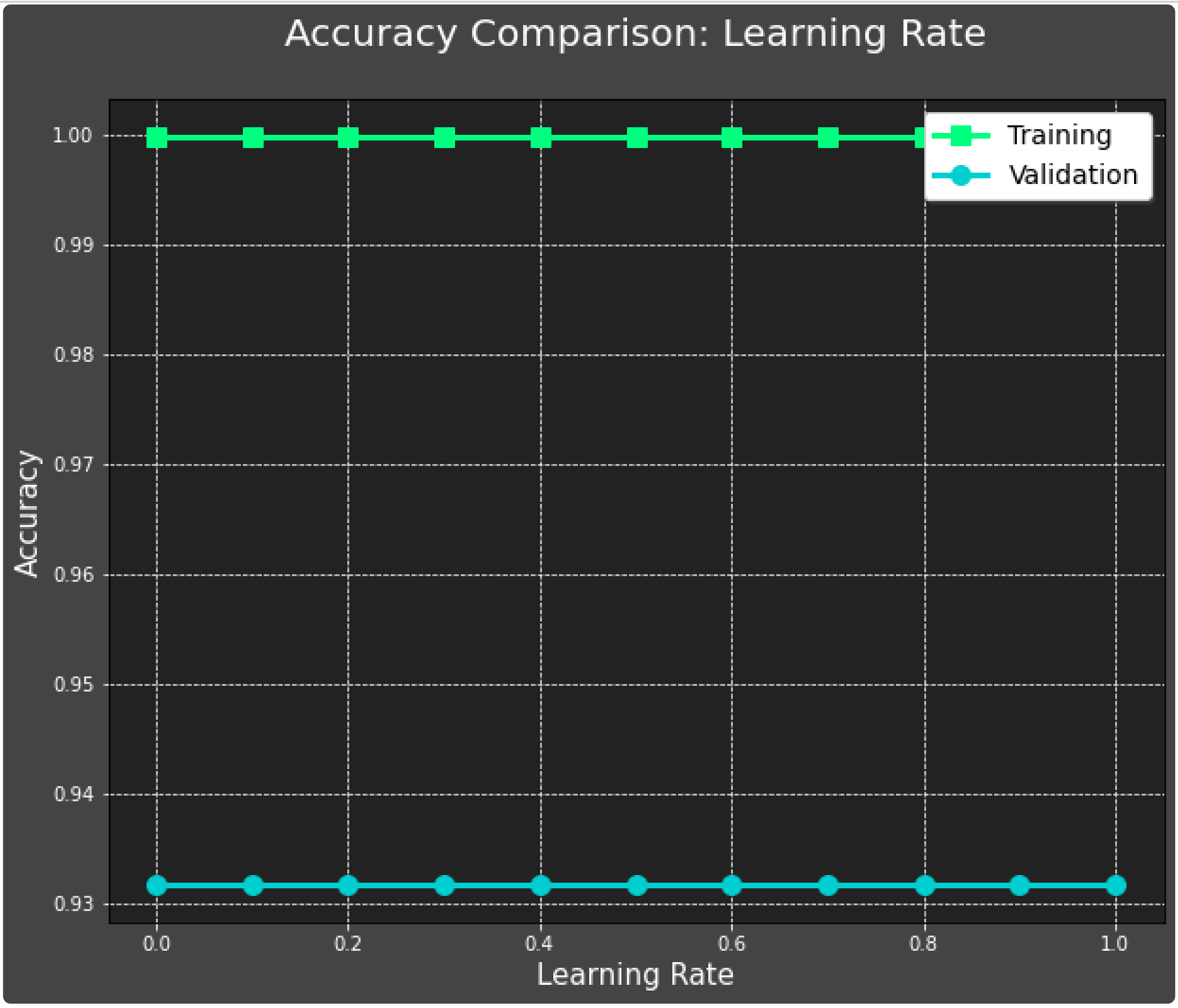
```
def learning_rate_accuracy(iterations, X_train, train_targets, X_val, val_targets, lear

  comparison_learning_rate = []
  learning_rate = learning_rate_start
  training_accuracy = 0
  validation_accuracy = 0

  for iteration in range(iterations):
      model = XGBRegressor(objective="reg:squarederror", n_jobs=-1, random_state=42, n_
      model.fit(X_train, train_targets)
      comparison_learning_rate.append([iteration, learning_rate, model.score(X_train, t
      print("XGBoost Regressor Model with 200 decision trees and max depth of 7:")
      print(f"Iteration number {iteration + 1} of {iterations} with learning rate of {l
      print(f"received a training score of {(model.score(X_train, train_targets)*100):
      learning_rate += learning_rate_increment
  comparison_learning_rate = pd.DataFrame(comparison_learning_rate, columns = ['iterati

  return(comparison_learning_rate)
```

```
learning_rate_accuracy = learning_rate_accuracy(11, train_inputs, train_targets, test_i
```

```
XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 1 of 11 with learning rate of 0.0:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 2 of 11 with learning rate of 0.1:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 3 of 11 with learning rate of 0.2:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 4 of 11 with learning rate of 0.30000000000000004:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 5 of 11 with learning rate of 0.4:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 6 of 11 with learning rate of 0.5:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 7 of 11 with learning rate of 0.6:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 8 of 11 with learning rate of 0.7:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 9 of 11 with learning rate of 0.7999999999999999:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 10 of 11 with learning rate of 0.8999999999999999:
received a training score of  99.99% and a validation score of  93.17%.

XGBoost Regressor Model with 200 decision trees and max depth of 7:
Iteration number 11 of 11 with learning rate of 0.9999999999999999:
```

received a training score of 99.99% and a validation score of 93.17%.

```
plot_one_param_accuracy(learning_rate_accuracy, key_param = "learning_rate", plot_title
                  x_label="Learning Rate", y_label="Accuracy", plot_y_lim = None,
                  train_color = rd.choice(greenish), val_color = rd.choice(blueish),
                  logscale_x = False)
```



`subsample` ⇨ RMSE comparison training vs validation (n_estimators is 200, max_depth is 7, and learning rate is 0.1 based on the testing above.)

```
def compare_subsample_rmse(iterations, subsample_start, subsample_increment):

  comparison_subsample = []
  subsample_count = subsample_start
  train_rmse = 0
  val_rmse = 0

  for iteration in range(iterations):
      train_rmse, val_rmse = test_parameters(subsample = subsample_count, n_estimators
      comparison_subsample.append([iteration, subsample_count, train_rmse, val_rmse])
      print(f"XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration {i
      print(f"received a training RMSE of {train_rmse: .2f} and a validation RMSE of {v
```

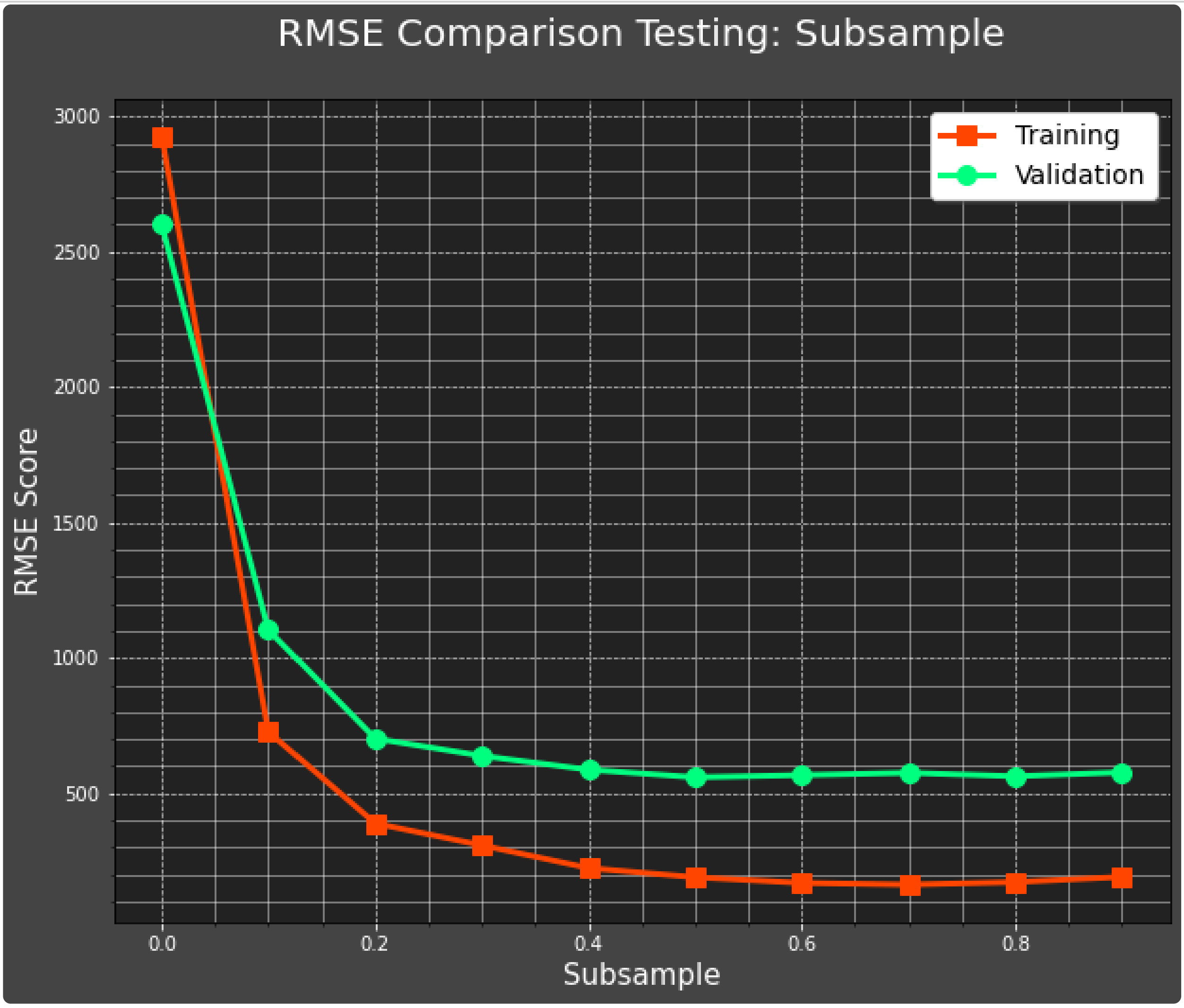
```
    subsample_count += subsample_increment

comparison_subsample = pd.DataFrame(comparison_subsample, columns = ['iteration', 'su
return(comparison_subsample)
```

```
subsample_rmse = compare_subsample_rmse(iterations=10, subsample_start=0, subsample_inc
```

```
XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 1 of 10 with
subsample of 0
received a training RMSE of  2927.22 and a validation RMSE of  2600.15.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 2 of 10 with
subsample of 0.1
received a training RMSE of  727.08 and a validation RMSE of  1103.44.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 3 of 10 with
subsample of 0.2
received a training RMSE of  388.64 and a validation RMSE of  701.99.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 4 of 10 with
subsample of 0.30000000000000004
received a training RMSE of  307.64 and a validation RMSE of  638.35.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 5 of 10 with
subsample of 0.4
received a training RMSE of  223.73 and a validation RMSE of  587.20.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 6 of 10 with
subsample of 0.5
received a training RMSE of  189.60 and a validation RMSE of  559.04.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 7 of 10 with
subsample of 0.6
received a training RMSE of  168.62 and a validation RMSE of  567.11.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 8 of 10 with
subsample of 0.7
received a training RMSE of  162.40 and a validation RMSE of  575.56.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 9 of 10 with
subsample of 0.7999999999999999
received a training RMSE of  171.66 and a validation RMSE of  562.78.

XGBoost(n_estimators=200, max_depth=7, learning_rate=0.1) on iteration 10 of 10 with
```

```
subsample of 0.8999999999999999
received a training RMSE of  190.84 and a validation RMSE of  577.32.
```

```
plot_one_param_rmse(subsample_rmse, key_param = 'subsample_count', plot_title="RMSE Com
                    x_label="Subsample", y_label='RMSE Score',
                    train_color = rd.choice(reddish), val_color = rd.choice(greenish),
                    logscale_x = False)
```



subsample ⇨ Accuracy comparison training vs validation (n_estimators is 200, max_depth is 7, and learning rate is 0.1 based on the testing above.)

```
def subsample_accuracy(iterations, X_train, train_targets, X_val, val_targets, subsampl

  comparison_subsample = []
  subsample = subsample_start
  training_accuracy = 0
  validation_accuracy = 0

  for iteration in range(iterations):
      model = XGBRegressor(objective="reg:squarederror", n_jobs=-1, random_state=42, n_
      model.fit(X_train, train_targets)
      comparison_subsample.append([iteration, subsample, model.score(X_train, train_tar
```

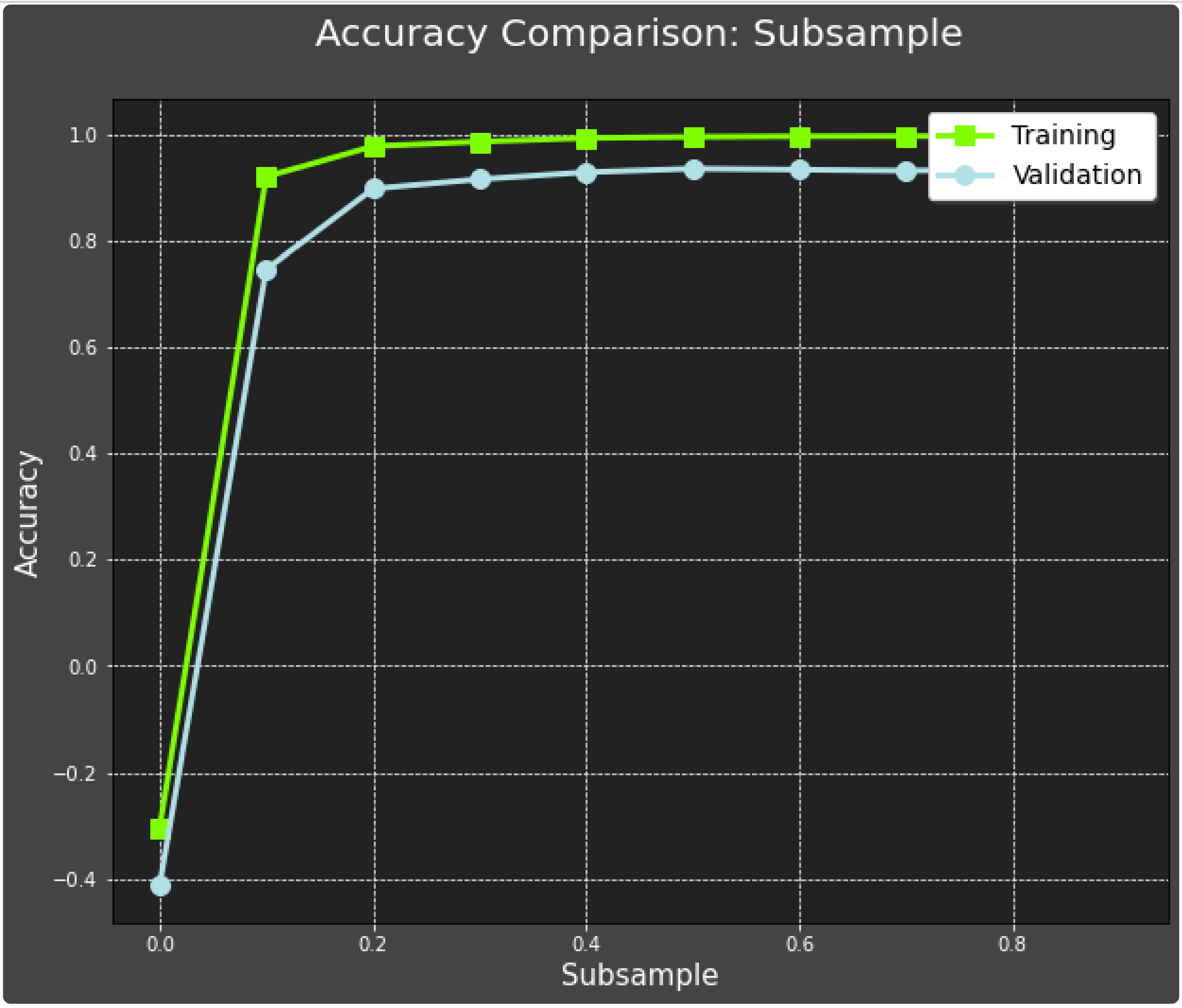
```
        print("XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1): \n")
        print(f"Iteration number {iteration + 1} of {iterations} with subsample of {subsa
        print(f"received a training score of {(model.score(X_train, train_targets)*100):
        subsample += subsample_increment
    comparison_subsample = pd.DataFrame(comparison_subsample, columns = ['iteration', 'su

    return(comparison_subsample)
```

```
subsample_accuracy = subsample_accuracy(10, train_inputs, train_targets, test_inputs, t
```

```
XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 1 of 10 with subsample of 0:
received a training score of -30.27% and a validation score of -41.13%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 2 of 10 with subsample of 0.1:
received a training score of  91.96% and a validation score of  74.58%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 3 of 10 with subsample of 0.2:
received a training score of  97.70% and a validation score of  89.71%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 4 of 10 with subsample of 0.30000000000000004:
received a training score of  98.56% and a validation score of  91.49%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 5 of 10 with subsample of 0.4:
received a training score of  99.24% and a validation score of  92.80%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 6 of 10 with subsample of 0.5:
received a training score of  99.45% and a validation score of  93.48%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):


Iteration number 7 of 10 with subsample of 0.6:
received a training score of  99.57% and a validation score of  93.29%.
```

```
XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):

Iteration number 8 of 10 with subsample of 0.7:
received a training score of  99.60% and a validation score of  93.08%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):

Iteration number 9 of 10 with subsample of 0.7999999999999999:
received a training score of  99.55% and a validation score of  93.39%.

XGBoostRegressor(n_estimators=200, max_depth=7, learning_rate=0.1):

Iteration number 10 of 10 with subsample of 0.8999999999999999:
received a training score of  99.45% and a validation score of  93.04%.
```

```
plot_one_param_accuracy(subsample_accuracy, key_param = "subsample", plot_title="Accura
                  x_label="Subsample", y_label="Accuracy", plot_y_lim = None,
                  train_color = rd.choice(greenish), val_color = rd.choice(blueish),
                  logscale_x = False)
```



MODEL PARAMETERS DECIDED:

XGBRegressor(n_estimators=200, max_depth=7, learning_rate=0.1, subsample=0.5)

```
xgbmodel_optimized = XGBRegressor(random_state=42, n_jobs=-1, n_estimators=200, max_dep
```

```
xgbmodel_optimized.fit(train_inputs, train_targets)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.1, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=7, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=200,
             n_jobs=-1, num_parallel_tree=1, predictor='auto', random_state=42,
             reg_alpha=0, reg_lambda=1, ...)
```

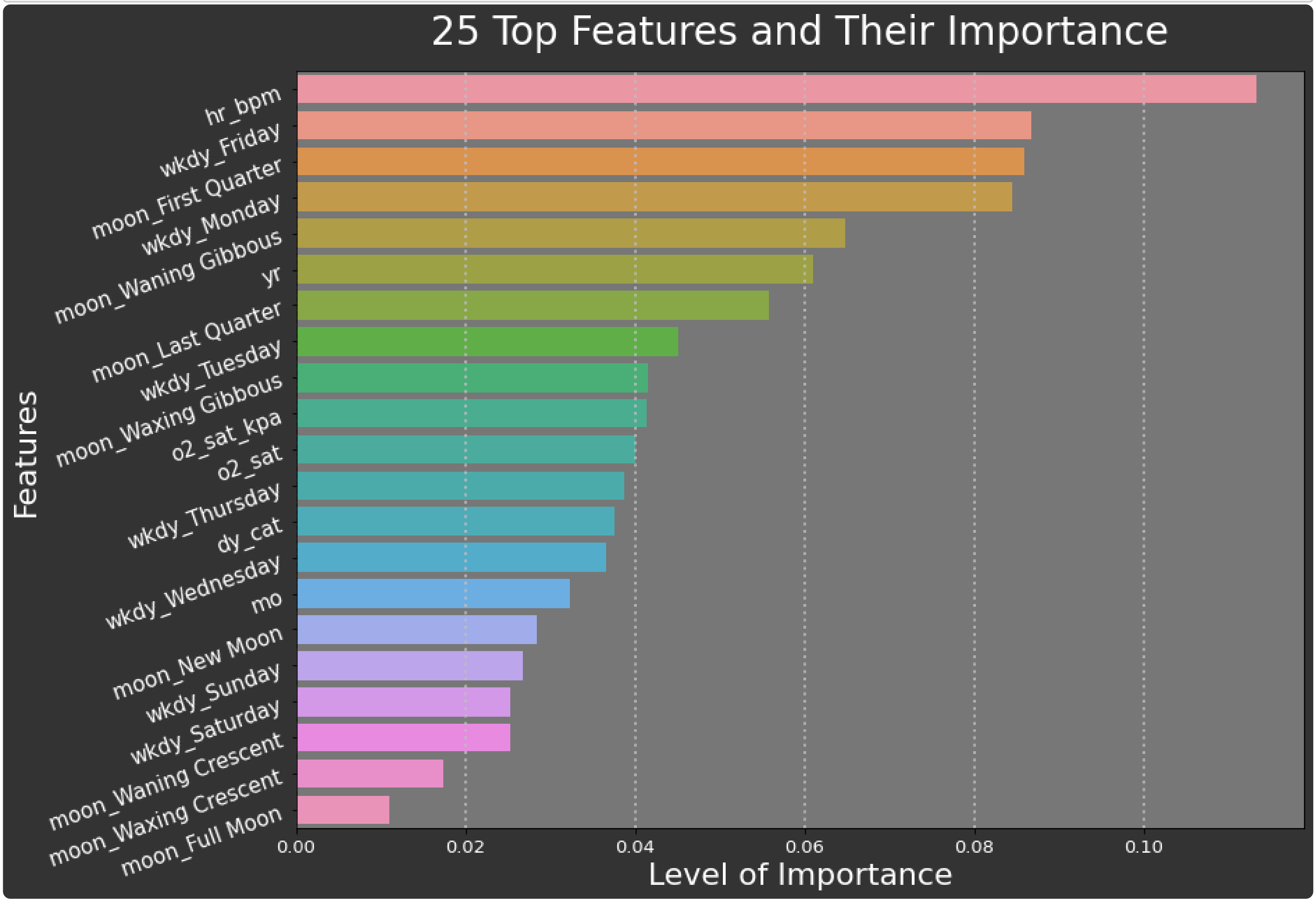
```
optimized_predictions = xgbmodel_optimized.predict(train_inputs)
```

```
feature_importances_df = xgboost_df[keeper_cols]

importance_df = pd.DataFrame({
    'feature': feature_importances_df.columns,
    'importance': xgbmodel_optimized.feature_importances_
}).sort_values('importance', ascending=False)
```

```
importance_df.head(10)
```

	feature	importance
3	hr_bpm	0.113305
14	wkdy_Friday	0.086774
6	moon_First Quarter	0.085905
15	wkdy_Monday	0.084472
11	moon_Waning Gibbous	0.064860
0	yr	0.061016
8	moon_Last Quarter	0.055773
19	wkdy_Tuesday	0.045100
13	moon_Waxing Gibbous	0.041568
5	o2_sat_kpa	0.041348

```
# Features and Their Importances Plotted
fig = plt.figure(figsize=(13, 10), facecolor='#333333')
ax = plt.axes(facecolor="#777777")
plt.xticks(fontsize=13, color="white")
plt.yticks(fontsize=13, color="white", rotation = 20, size=15)
plt.grid(axis='both', which='major', color='#bfbec2', linestyle=':', linewidth=2, alpha
plt.grid(axis="both", which='minor', color='#bfbec2', linestyle='-', alpha=0.4)
sns.barplot(data=importance_df.head(25), x='importance', y='feature', alpha=1)
sns.set_palette(rd.choice(color_groups))
ax.set_title("25 Top Features and Their Importance", fontsize=28, color="white", pad=20
ax.set_xlabel("Level of Importance", fontsize=22, color="white")
ax.set_ylabel("Features", fontsize=22, color="white")
fig.show()
plt.show()
```



Final optimized model's training and validation scores

```
scores = xgbmodel_optimized.score(train_inputs, train_targets), xgbmodel_optimized.scor
```

```
print(f'Optimized model training accuracy score is {scores[0] * 100 :.2f}%, \nand valic
```

```
Optimized model training accuracy score is 99.45%,
and validation score of 93.48%. Not too shabby!
```