**Taxi Ride Demand Prediction Code**

import polars as pl

import numpy as np

import math

import statistics as stat

from lets\_plot import \*

from lets\_plot.mapping import as\_discrete

from sklearn import model\_selection

from sklearn.metrics import mean\_squared\_error

from sklearn.linear\_model import LinearRegression

LetsPlot.setup\_html()

df = pl.read\_csv("dataset.csv", parse\_dates = True).drop("ID")

df = df.with\_row\_count(name = "Time", offset = 0)

df.shape

df.describe()

df\_c1 = df.filter(pl.col("Junction") == 1)

df\_c2 = df.filter(pl.col("Junction") == 2)

df\_c3 = df.filter(pl.col("Junction") == 3)

df\_c4 = df.filter(pl.col("Junction") == 4)

def df\_splitter(df):

df\_train = df.filter(pl.col("DateTime") < pl.datetime(2021, 6, 1))

df\_valid = df.filter(pl.col("DateTime") >= pl.datetime(2021, 6, 1))

return df\_train, df\_valid

df\_c1\_train, df\_c1\_valid = df\_splitter(df\_c1)

df\_c2\_train, df\_c2\_valid = df\_splitter(df\_c2)

df\_c3\_train, df\_c3\_valid = df\_splitter(df\_c3)

df\_c4\_train, df\_c4\_valid = df\_splitter(df\_c4)

c1\_color = 'blue'

c2\_color = 'green'

c3\_color = 'red'

c4\_color = 'black'

plt\_ts\_c1 = \

ggplot(df\_c1\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles"),

color = c1\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Vehicles", title = "No of Taxis booked in Manhatten, New York")

plt\_ts\_c2 = \

ggplot(df\_c2\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles"),

color = c2\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Vehicles", title = "No of Taxis booked in Los Angels, California")

plt\_ts\_c3 = \

ggplot(df\_c3\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles"),

color = c3\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Vehicles", title = "No of Taxis booked in New Jersey City, New Jersey")

plt\_ts\_c4 = \

ggplot(df\_c4\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles"),

color = c4\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Vehicles", title = "No of Taxis booked in Dallas, Texas")

ts\_plts = GGBunch()

ts\_plts.add\_plot(plt\_ts\_c1, 0, 0, 800, 300)

ts\_plts.add\_plot(plt\_ts\_c2, 0, 300, 800, 300)

ts\_plts.add\_plot(plt\_ts\_c3, 0, 600, 800, 300)

ts\_plts.add\_plot(plt\_ts\_c4, 0, 900, 800, 300)

ts\_plts

df\_c1\_train = df\_c1\_train.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c2\_train = df\_c2\_train.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c3\_train = df\_c3\_train.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c4\_train = df\_c4\_train.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c1\_train = df\_c1\_train.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c2\_train = df\_c2\_train.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c3\_train = df\_c3\_train.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c4\_train = df\_c4\_train.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c1\_train = df\_c1\_train.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c2\_train = df\_c2\_train.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c3\_train = df\_c3\_train.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c4\_train = df\_c4\_train.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c1\_train = df\_c1\_train.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c2\_train = df\_c2\_train.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c3\_train = df\_c3\_train.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c4\_train = df\_c4\_train.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c1\_train = df\_c1\_train.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c2\_train = df\_c2\_train.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c3\_train = df\_c3\_train.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c4\_train = df\_c4\_train.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

def mean\_vehicles(df) -> pl.Expr:

return pl.col("Vehicles").mean()

df\_train = pl.concat([df\_c1\_train, df\_c2\_train, df\_c3\_train, df\_c4\_train])

df\_monthly = (

df\_train.groupby(["Junction", "Month"])

.agg([mean\_vehicles("Month")])

.sort("Junction")

)

df\_day\_week = (

df\_train.groupby(["Junction", "Day\_week"])

.agg([mean\_vehicles("Day\_week")])

.sort("Junction")

)

df\_day\_month = (

df\_train.groupby(["Junction", "Day\_month"])

.agg([mean\_vehicles("Day\_month")])

.sort("Junction")

)

df\_hourly = (

df\_train.groupby(["Junction", "Hour"])

.agg([mean\_vehicles("Hour")])

.sort("Junction")

)

plt\_monthly = \

ggplot(df\_monthly)+\

geom\_line(aes(x = "Month", y = "Vehicles", color = as\_discrete("Junction")), size = 3)+\

scale\_x\_discrete(breaks = list(range(1,13,1)))+\

scale\_color\_manual(values = [c1\_color, c2\_color, c3\_color, c4\_color])+\

theme\_bw()+\

labs(x = "Month", y = "No. of Taxis booked", title = "Average No. of Taxis booked in a Year")

plt\_day\_week = \

ggplot(df\_day\_week)+\

geom\_line(aes(x = "Day\_week", y = "Vehicles", color = as\_discrete("Junction")), size = 3)+\

scale\_x\_discrete(breaks = list(range(1,8,1)))+\

scale\_color\_manual(values = [c1\_color, c2\_color, c3\_color, c4\_color])+\

theme\_bw()+\

labs(x = "Day of the Week", y = "No. of Taxis booked", title = "Average No. of Taxis booked in a Week(Day of a week)")

plt\_day\_month = \

ggplot(df\_day\_month)+\

geom\_line(aes(x = "Day\_month", y = "Vehicles", color = as\_discrete("Junction")), size = 3)+\

scale\_x\_discrete(breaks = list(range(1,32,1)))+\

scale\_color\_manual(values = [c1\_color, c2\_color, c3\_color, c4\_color])+\

theme\_bw()+\

labs(x = "Day of the Month", y = "No. of Taxis booked", title = "Average No. of Taxis booked in a Month(Day of a month)")

plt\_hourly = \

ggplot(df\_hourly)+\

geom\_line(aes(x = "Hour", y = "Vehicles", color = as\_discrete("Junction")), size = 3)+\

scale\_color\_manual(values = [c1\_color, c2\_color, c3\_color, c4\_color])+\

theme\_bw()+\

labs(x = "Hour", y = "No. of Taxis booked", title = "Average No. of Taxis booked in a Day")

ts\_plts\_2 = GGBunch()

ts\_plts\_2.add\_plot(plt\_monthly, 0, 0, 900, 300)

ts\_plts\_2.add\_plot(plt\_day\_week, 0, 300, 900, 300)

ts\_plts\_2.add\_plot(plt\_day\_month, 0, 600, 900, 300)

ts\_plts\_2.add\_plot(plt\_hourly, 0, 900, 900, 300)

ts\_plts\_2

df\_c1\_train = df\_c1\_train.with\_columns([

(pl.when(pl.col("Day\_week") == 7)

.then(1)

.when(pl.col("Day\_week") == 6)

.then(1)

.otherwise(0))

.alias("Weekend")

])

df\_c2\_train = df\_c2\_train.with\_columns([

(pl.when(pl.col("Day\_week") == 7)

.then(1)

.when(pl.col("Day\_week") == 6)

.then(1)

.otherwise(0))

.alias("Weekend")

])

df\_c1\_train = df\_c1\_train.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c2\_train = df\_c2\_train.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c3\_train = df\_c3\_train.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c4\_train = df\_c4\_train.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c1\_train = df\_c1\_train.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c2\_train = df\_c2\_train.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c3\_train = df\_c3\_train.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c4\_train = df\_c4\_train.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c1\_train = df\_c1\_train.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c2\_train = df\_c2\_train.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c3\_train = df\_c3\_train.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c4\_train = df\_c4\_train.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c1\_train = df\_c1\_train.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

df\_c2\_train = df\_c2\_train.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

df\_c3\_train = df\_c3\_train.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

df\_c4\_train = df\_c4\_train.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

plt\_ts\_c1\_log = \

ggplot(df\_c1\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_log"),

color = c1\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Log Vehicles", title = "No. of Taxis at Manhatten(After Log Transform)")

plt\_ts\_c2\_log = \

ggplot(df\_c2\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_log"),

color = c2\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Log Vehicles", title = "No. of Taxis at Los Angels(After Log Transform)")

plt\_ts\_c3\_log = \

ggplot(df\_c3\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_log"),

color = c3\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Log Vehicles", title = "No. of Taxis at New Jersey City (After Log Transform)")

plt\_ts\_c4\_log = \

ggplot(df\_c4\_train)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_log"),

color = c4\_color, sampling = "none")+\

scale\_x\_datetime(format = "%b %Y")+\

theme\_bw()+\

labs(x = "Date", y = "Log Vehicles", title = "No. of Taxis at Dallas (After Log Transform)")

ts\_plts\_log = GGBunch()

ts\_plts\_log.add\_plot(plt\_ts\_c1\_log, 0, 0, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c1, 500, 0, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c2\_log, 0, 320, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c2, 500, 320, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c3\_log, 0, 640, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c3, 500, 640, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c4\_log, 0, 960, 500, 300)

ts\_plts\_log.add\_plot(plt\_ts\_c4, 500, 960, 500, 300)

ts\_plts\_log

df\_c1\_valid = df\_c1\_valid.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c2\_valid = df\_c2\_valid.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c3\_valid = df\_c3\_valid.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c4\_valid = df\_c4\_valid.with\_columns(pl.col("DateTime").dt.year().alias("Year"))

df\_c1\_valid = df\_c1\_valid.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c2\_valid = df\_c2\_valid.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c3\_valid = df\_c3\_valid.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c4\_valid = df\_c4\_valid.with\_columns(pl.col("DateTime").dt.month().alias("Month"))

df\_c1\_valid = df\_c1\_valid.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c2\_valid = df\_c2\_valid.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c3\_valid = df\_c3\_valid.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c4\_valid = df\_c4\_valid.with\_columns(pl.col("DateTime").dt.day().alias("Day\_month"))

df\_c1\_valid = df\_c1\_valid.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c2\_valid = df\_c2\_valid.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c3\_valid = df\_c3\_valid.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c4\_valid = df\_c4\_valid.with\_columns(pl.col("DateTime").dt.weekday().alias("Day\_week"))

df\_c1\_valid = df\_c1\_valid.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c2\_valid = df\_c2\_valid.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c3\_valid = df\_c3\_valid.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c4\_valid = df\_c4\_valid.with\_columns(pl.col("DateTime").dt.hour().alias("Hour"))

df\_c1\_valid = df\_c1\_valid.with\_columns([

(pl.when(pl.col("Day\_week") == 7)

.then(1)

.when(pl.col("Day\_week") == 6)

.then(1)

.otherwise(0))

.alias("Weekend")

])

df\_c2\_valid = df\_c2\_valid.with\_columns([

(pl.when(pl.col("Day\_week") == 7)

.then(1)

.when(pl.col("Day\_week") == 6)

.then(1)

.otherwise(0))

.alias("Weekend")

])

df\_c1\_valid = df\_c1\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c2\_valid = df\_c2\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c3\_valid = df\_c3\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c4\_valid = df\_c4\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 0) & (pl.col("Hour") <= 5))

.then(1)

.otherwise(0))

.alias("Mid\_to\_five")

])

df\_c1\_valid = df\_c1\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c2\_valid = df\_c2\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c3\_valid = df\_c3\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c4\_valid = df\_c4\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 5) & (pl.col("Hour") <= 12))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c1\_valid = df\_c1\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c2\_valid = df\_c2\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c3\_valid = df\_c3\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c4\_valid = df\_c4\_valid.with\_columns([

(pl.when((pl.col("Hour") >= 12) & (pl.col("Hour") <= 0))

.then(1)

.otherwise(0))

.alias("Five\_to\_noon")

])

df\_c1\_valid = df\_c1\_valid.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

df\_c2\_valid = df\_c2\_valid.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

df\_c3\_valid = df\_c3\_valid.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

df\_c4\_valid = df\_c4\_valid.with\_columns([

(pl.col("Vehicles").log()).alias("Vehicles\_log")

])

def objective(trial):

xtrain = df\_train.drop(["DateTime", "Junction", "Vehicles", "Vehicles\_log"]).to\_numpy()

xvalid = df\_valid.drop(["DateTime", "Junction", "Vehicles", "Vehicles\_log"]).to\_numpy()

ytrain = df\_train.get\_column("Vehicles\_log").to\_numpy()

yvalid = df\_valid.get\_column("Vehicles\_log").to\_numpy()

reg\_model = LinearRegression().fit(xtrain, ytrain)

reg\_preds\_train = reg\_model.predict(xtrain)

reg\_preds\_valid = reg\_model.predict(xvalid)

reg\_resid\_train = (ytrain - reg\_preds\_train)

reg\_resid\_valid = (yvalid - reg\_preds\_valid)

params = {'objective': 'reg:squarederror',

'eval\_metric': 'rmse',

'seed': 19970507,

'eta': trial.suggest\_float("eta", 1e-2, 0.25, log = True),

'max\_depth': trial.suggest\_int("max\_depth", 1, 7),

'lambda': trial.suggest\_float("lambda", 1e-8, 100.0, log = True),

'alpha': trial.suggest\_float("alpha", 1e-8, 100.0, log = True),

}

dmat\_train = xgb.DMatrix(xtrain, label = reg\_resid\_train)

dmat\_valid = xgb.DMatrix(xvalid, label = reg\_resid\_valid)

watchlist = [(dmat\_train, 'train'), (dmat\_valid, 'eval')]

xgb\_model = xgb.train(params,

dtrain = dmat\_train,

num\_boost\_round = trial.suggest\_int("num\_boost\_round", 20, 3000),

evals = watchlist,

verbose\_eval = False)

xgb\_preds\_valid = xgb\_model.predict(dmat\_valid)

preds = (reg\_preds\_valid + xgb\_preds\_valid)

return math.sqrt(mean\_squared\_error(yvalid, preds))

import optuna

import xgboost as xgb

optuna.logging.set\_verbosity(optuna.logging.WARNING) # Suppress log messages

df\_train = df\_c1\_train

df\_valid = df\_c1\_valid

study\_c1 = optuna.create\_study(direction = 'minimize')

study\_c1.optimize(objective, n\_trials = 5)

df\_train = df\_c2\_train

df\_valid = df\_c2\_valid

study\_c2 = optuna.create\_study(direction = 'minimize')

study\_c2.optimize(objective, n\_trials = 5)

df\_train = df\_c3\_train

df\_valid = df\_c3\_valid

study\_c3 = optuna.create\_study(direction = 'minimize')

study\_c3.optimize(objective, n\_trials = 5)

df\_train = df\_c4\_train

df\_valid = df\_c4\_valid

study\_c4 = optuna.create\_study(direction = 'minimize')

study\_c4.optimize(objective, n\_trials = 5)

import optuna

def final\_model\_preds(optuna\_params, df\_train, df\_valid, junction):

xtrain = df\_train.drop(["DateTime", "Junction", "Vehicles", "Vehicles\_log"]).to\_numpy()

xvalid = df\_valid.drop(["DateTime", "Junction", "Vehicles", "Vehicles\_log"]).to\_numpy()

ytrain = df\_train.get\_column("Vehicles\_log").to\_numpy()

yvalid = df\_valid.get\_column("Vehicles\_log").to\_numpy()

yvalid\_orig = df\_valid.get\_column("Vehicles").to\_numpy()

reg\_model = LinearRegression().fit(xtrain, ytrain)

reg\_preds\_train = reg\_model.predict(xtrain)

reg\_preds\_valid = reg\_model.predict(xvalid)

reg\_resid\_train = (ytrain - reg\_preds\_train)

reg\_resid\_valid = (yvalid - reg\_preds\_valid)

best\_params = {'objective': 'reg:squarederror',

'eval\_metric': 'rmse',

'seed': 19970507,

'eta': optuna\_params['eta'],

'max\_depth': optuna\_params['max\_depth'],

'lambda': optuna\_params['lambda'],

'alpha': optuna\_params['alpha'],

}

dmat\_train = xgb.DMatrix(xtrain, label = reg\_resid\_train)

dmat\_valid = xgb.DMatrix(xvalid, label = reg\_resid\_valid)

watchlist = [(dmat\_train, 'train'), (dmat\_valid, 'eval')]

xgb\_model = xgb.train(best\_params,

dtrain = dmat\_train,

num\_boost\_round = optuna\_params['num\_boost\_round'],

evals = watchlist,

early\_stopping\_rounds = 100,

verbose\_eval = False)

xgb\_preds\_valid = xgb\_model.predict(dmat\_valid)

preds = (reg\_preds\_valid + xgb\_preds\_valid)

preds\_orig = [math.exp(x) for x in preds]

rmse = math.sqrt(mean\_squared\_error(yvalid\_orig, preds\_orig))

return preds\_orig, rmse

preds\_c1, valid\_rmse\_c1 = final\_model\_preds(optuna\_params = study\_c1.best\_params,

df\_train = df\_c1\_train,

df\_valid = df\_c1\_valid,

junction = 1)

preds\_c2, valid\_rmse\_c2 = final\_model\_preds(optuna\_params = study\_c2.best\_params,

df\_train = df\_c2\_train,

df\_valid = df\_c2\_valid,

junction = 2)

preds\_c3, valid\_rmse\_c3 = final\_model\_preds(optuna\_params = study\_c3.best\_params,

df\_train = df\_c3\_train,

df\_valid = df\_c3\_valid,

junction = 3)

preds\_c4, valid\_rmse\_c4 = final\_model\_preds(optuna\_params = study\_c4.best\_params,

df\_train = df\_c4\_train,

df\_valid = df\_c4\_valid,

junction = 4)

df\_c1\_labels = pl.DataFrame(

{'DateTime': df\_c1\_valid.get\_column("DateTime"),

'Vehicles': df\_c1\_valid.get\_column("Vehicles"),

'Group': ["Label"]\*len(df\_c1\_valid)}

)

df\_c1\_preds = pl.DataFrame(

{'DateTime\_preds': df\_c1\_valid.get\_column("DateTime"),

'Vehicles\_preds': preds\_c1,

'Group\_preds': ["Predictions"]\*len(df\_c1\_valid)}

)

df\_c1 = (

pl.concat([df\_c1\_labels, df\_c1\_preds], how = 'horizontal')

.with\_columns(

(pl.lit("True Values").alias("Group\_label")),

(pl.lit("Predictions").alias("Group\_pred")))

)

plt\_c1 = \

ggplot(df\_c1)+\

geom\_line(aes(x = "DateTime", y = "Vehicles", color = "Group\_label"),

sampling = "none", size = 0.5, show\_legend = True)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_preds", color = "Group\_pred"),

sampling = "none", size = 0.5, show\_legend = True)+\

scale\_color\_manual(values = ['white', c1\_color])+\

scale\_x\_datetime(format = "%Y-%m-%d")+\

scale\_y\_continuous(limits = [20, 145])+\

theme\_bw()+\

labs(x = "Date", y = "No. of Taxis", title = "Predicted count of taxis in in Manhatten")

df\_c2\_labels = pl.DataFrame(

{'DateTime': df\_c2\_valid.get\_column("DateTime"),

'Vehicles': df\_c2\_valid.get\_column("Vehicles"),

'Group': ["Label"]\*len(df\_c2\_valid)}

)

df\_c2\_preds = pl.DataFrame(

{'DateTime\_preds': df\_c2\_valid.get\_column("DateTime"),

'Vehicles\_preds': preds\_c2,

'Group\_preds': ["Predictions"]\*len(df\_c2\_valid)}

)

df\_c2 = (

pl.concat([df\_c2\_labels, df\_c2\_preds], how = 'horizontal')

.with\_columns(

(pl.lit("True Values").alias("Group\_label")),

(pl.lit("Predictions").alias("Group\_pred")))

)

plt\_c2 = \

ggplot(df\_c2)+\

geom\_line(aes(x = "DateTime", y = "Vehicles", color = "Group\_label"),

sampling = "none", size = 0.5, show\_legend = True)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_preds", color = "Group\_pred"),

sampling = "none", size = 0.5, show\_legend = True)+\

scale\_color\_manual(values = ['white', c2\_color])+\

scale\_x\_datetime(format = "%Y-%m-%d")+\

scale\_y\_continuous(limits = [20, 50])+\

theme\_bw()+\

labs(x = "Date", y = "No. of Taxis", title = "Predicted count of taxis in in Los Angels")

df\_c3\_labels = pl.DataFrame(

{'DateTime': df\_c3\_valid.get\_column("DateTime"),

'Vehicles': df\_c3\_valid.get\_column("Vehicles"),

'Group': ["Label"]\*len(df\_c3\_valid)}

)

df\_c3\_preds = pl.DataFrame(

{'DateTime\_preds': df\_c3\_valid.get\_column("DateTime"),

'Vehicles\_preds': preds\_c3,

'Group\_preds': ["Predictions"]\*len(df\_c3\_valid)}

)

df\_c3 = (

pl.concat([df\_c3\_labels, df\_c3\_preds], how = 'horizontal')

.with\_columns(

(pl.lit("True Values").alias("Group\_label")),

(pl.lit("Predictions").alias("Group\_pred")))

)

plt\_c3 = \

ggplot(df\_c3)+\

geom\_line(aes(x = "DateTime", y = "Vehicles", color = "Group\_label"),

sampling = "none", size = 0.5, show\_legend = True)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_preds", color = "Group\_pred"),

sampling = "none", size = 0.5, show\_legend = True)+\

scale\_color\_manual(values = ['white', c3\_color])+\

scale\_x\_datetime(format = "%Y-%m-%d")+\

scale\_y\_continuous(limits = [20, 120])+\

theme\_bw()+\

labs(x = "Date", y = "No. of Taxis", title = "Predicted count of taxis in New Jersey City")

df\_c4\_labels = pl.DataFrame(

{'DateTime': df\_c4\_valid.get\_column("DateTime"),

'Vehicles': df\_c4\_valid.get\_column("Vehicles"),

'Group': ["Label"]\*len(df\_c4\_valid)}

)

df\_c4\_preds = pl.DataFrame(

{'DateTime\_preds': df\_c4\_valid.get\_column("DateTime"),

'Vehicles\_preds': preds\_c4,

'Group\_preds': ["Predictions"]\*len(df\_c4\_valid)}

)

df\_c4 = (

pl.concat([df\_c4\_labels, df\_c4\_preds], how = 'horizontal')

.with\_columns(

(pl.lit("True Values").alias("Group\_label")),

(pl.lit("Predictions").alias("Group\_pred")))

)

plt\_c4 = \

ggplot(df\_c4)+\

geom\_line(aes(x = "DateTime", y = "Vehicles", color = "Group\_label"),

sampling = "none", size = 0.5, show\_legend = True)+\

geom\_line(aes(x = "DateTime", y = "Vehicles\_preds", color = "Group\_pred"),

sampling = "none", size = 0.5, show\_legend = True)+\

scale\_color\_manual(values = ['white', c4\_color])+\

scale\_x\_datetime(format = "%Y-%m-%d")+\

scale\_y\_continuous(limits = [0, 40])+\

theme\_bw()+\

labs(x = "Date", y = "No. of Taxis", title = "Predicted count of taxis in Dallas City")

#results\_plts = GGBunch()

#results\_plts.add\_plot(plt\_c1, 0, 0, 900, 300)

#results\_plts.add\_plot(plt\_c2, 0, 300, 900, 300)

#results\_plts.add\_plot(plt\_c3, 0, 600, 900, 300)

#results\_plts.add\_plot(plt\_c4, 0, 900, 900, 300)

#results\_plts

import tkinter as tk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.pyplot as plt

import pandas as pd

from tkinter import PhotoImage

from plotnine import \*

from PIL import Image, ImageTk

# Define the function to plot the graph

def man():

fig, ax = plt.subplots()

#ax.plot(df\_c1.get\_column('DateTime'), df\_c1.get\_column('Vehicles'), color='white')

ax.plot(df\_c1.get\_column('DateTime\_preds'), preds\_c1, color=c1\_color, label='Predictions')

ax.set\_xlabel('Date')

ax.set\_ylabel('No. of Taxis')

ax.set\_title('Predicted count of taxis in Manhattan')

ax.legend()

# Set the x-tick labels to your desired dates

dates = ['2023-05-04', '2023-05-07', '2023-05-12', '2023-05-16', '2023-05-21', '2023-05-24', '2023-05-28', '2023-05-31']

ax.set\_xticklabels(dates)

# Create a GUI window

root = tk.Tk()

root.title('Prediction of the plot for the city1(Manhattan)')

# Embed the plot in the GUI

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Create a button to close the window

button = tk.Button(root, text='Close', command=root.quit)

button.pack()

# Run the GUI

tk.mainloop()

#----------------------------------------------------------------------------------

def la():

fig, ax = plt.subplots()

#ax.plot(df\_c2.get\_column('DateTime'), df\_c2.get\_column('Vehicles'), color='white')

ax.plot(df\_c2.get\_column('DateTime\_preds'), preds\_c2, color=c2\_color, label='Predictions')

ax.set\_xlabel('Date')

ax.set\_ylabel('No. of Taxis')

ax.set\_title('Predicted count of taxis in Los Angels')

ax.legend()

# Set the x-tick labels to your desired dates

dates = ['2023-05-04', '2023-05-07', '2023-05-12', '2023-05-16', '2023-05-21', '2023-05-24', '2023-05-28', '2023-05-31']

ax.set\_xticklabels(dates)

# Create a GUI window

root = tk.Tk()

root.title('Prediction of the plot for the city2(Los Angels)')

# Embed the plot in the GUI

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Create a button to close the window

button = tk.Button(root, text='Close', command=root.quit)

button.pack()

# Run the GUI

tk.mainloop()

#----------------------------------------------------------------------------------

def nj():

fig, ax = plt.subplots()

#ax.plot(df\_c3.get\_column('DateTime'), df\_c3.get\_column('Vehicles'), color='white')

ax.plot(df\_c3.get\_column('DateTime\_preds'), preds\_c3, color=c3\_color, label='Predictions')

ax.set\_xlabel('Date')

ax.set\_ylabel('No. of Taxis')

ax.set\_title('Predicted count of taxis in New Jersey City')

ax.legend()

# Set the x-tick labels to your desired dates

dates = ['2023-05-04', '2023-05-07', '2023-05-12', '2023-05-16', '2023-05-21', '2023-05-24', '2023-05-28', '2023-05-31']

ax.set\_xticklabels(dates)

# Create a GUI window

root = tk.Tk()

root.title('Prediction of the plot for the city3(New Jersey)')

# Embed the plot in the GUI

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Create a button to close the window

button = tk.Button(root, text='Close', command=root.quit)

button.pack()

# Run the GUI

tk.mainloop()

#----------------------------------------------------------------------------------

def da():

fig, ax = plt.subplots()

#ax.plot(df\_c4.get\_column('DateTime'), df\_c4.get\_column('Vehicles'), color='white')

ax.plot(df\_c4.get\_column('DateTime\_preds'), preds\_c4, color=c4\_color, label='Predictions')

ax.set\_xlabel('Date')

ax.set\_ylabel('No. of Taxis')

ax.set\_title('Predicted count of taxis count in Dallas')

ax.legend()

# Set the x-tick labels to your desired dates

dates = ['2023-05-04', '2023-05-07', '2023-05-12', '2023-05-16', '2023-05-21', '2023-05-24', '2023-05-28', '2023-05-31']

ax.set\_xticklabels(dates)

# Create a GUI window

root = tk.Tk()

root.title('Prediction of the plot for the city4(Dallas)')

# Embed the plot in the GUI

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Create a button to close the window

button = tk.Button(root, text='Close', command=root.quit)

button.pack()

# Run the GUI

tk.mainloop()

#----------------------------------------------------------------------------------

# Create the GUI

root = tk.Tk()

root.title('Taxi Ride Demand Prediction')

heading\_label = tk.Label(root, text='Taxi Ride Demand Prediction Application', font=('Arial', 20, 'bold'))

heading\_label.pack()

# Load the image

image = PhotoImage(file='image.png')

#Create a label widget to display the image

label = tk.Label(root, image=image)

label.pack()

#--------------------------------------------------------------------

#-----------------------------------------------------------

# Create a button to plot the graph

button = tk.Button(root, text='Predict the plot for the city1(Manhatten)', command=man)

button.pack()

button = tk.Button(root, text='Predict the plot for the city2(Los Angels)', command=la)

button.pack()

button = tk.Button(root, text='Predict the plot for the city3(New Jersey)', command=nj)

button.pack()

button = tk.Button(root, text='Predict the plot for the city4(Dallas)', command=da)

button.pack()

# Run the GUI

root.mainloop()