```
###Certain Data and names redacted for confidentiality###

#imports
import pandas as pd
#read csv, create dataframes

import numpy as np
#used for vector values

import matplotlib.pyplot as plt
#Each pyplot function makes some change to a figure: e.g., creates a figur
e, creates a plotting area in a figure,
#plots some lines in a plotting area, decorates the plot with labels, etc.

from sklearn.linear_model import LinearRegression
#check features for best fit

from sklearn.linear_model import LogisticRegression
#check features for best fit

from sklearn.metrics import mean_absolute_error
#check accuracy of model

from sklearn.model_selection import train_test_split
#used to create training data and validation data

from sklearn.tree import DecisionTreeRegressor
#Decision tree builds regression or classification models in the form of a
 tree structure. It breaks down a dataset into
#smaller and smaller subsets while at the same time an associated decision
 tree is incrementally developed. The final
#result is a tree with decision nodes and leaf nodes.

from sklearn.ensemble import RandomForestRegressor
#Random forest builds multiple decision trees and merges them together to
get a more accurate and stable prediction.
#Random forest has nearly the same hyperparameters as a decision tree or a
 bagging classifier. ... Random forest adds additional randomness to the m
odel, while growing the trees.

from sklearn.preprocessing import StandardScaler
#used to Standardize features by removing the mean and scaling to unit var
iance

from random import sample
```

```
from random import shuffle
from random import Random
print("Ready to Roll")


#import data and create dataframe
#sample set to a set random sample of 1000 instances
#df = pd.read_csv('**********')
#df = df.sample(1000, random_state=42)
df = pd.read_csv("******************")
#label encoder to allow use of the mpn column
gle = LabelEncoder()

#adding lable column which groups the mfr by labels
df['label'] = df.groupby(pd.Grouper(key='mfr')).ngroup()

#creating labels for mpn
mpn_labels = np.unique(df['mpn'])
mpn_labels = gle.fit_transform(df['mpn'])
mpn_mapping = {index: label for index, label in
                  enumerate(gle.classes_)}

#adding mpn labels to the dataframe
df['mpn_labels']= mpn_labels

#features to be used in the model after analyzing all features to find the
 optimal result
clean_df_cols = [************************]

#creating data frame from the features
clean_df = df[clean_df_cols]


#simplifying the X and y variables
X = clean_df

y = df.expected_leadtime
```

```
#function of how labels for the mfr column are grouped
def label(DF,label):
        Y = DF.groupby(["label"]).get_group(label)
        Input = pd.DataFrame(Y)
        pd.DataFrame(Input)
        return Input

train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 42,
 test_size=0.20)
# split data into training and validation data, for both features and targ
et
# The split is based on a random number generator. Supplying a numeric val
ue to
# the random_state argument guarantees we get the same split every time we
# run this script.

forest_model = RandomForestRegressor(random_state=42)
forest_model.fit(train_X, train_y)
Expect_Time_preds = forest_model.predict(val_X)
print(mean_absolute_error(val_y, Expect_Time_preds))


# Define the models
model_1 = RandomForestRegressor(n_estimators=50, random_state=42)
model_2 = RandomForestRegressor(n_estimators=100, random_state=42)
model_3 = RandomForestRegressor(n_estimators=100, criterion='mae', random_
state=42)
model_4 = RandomForestRegressor(n_estimators=200, min_samples_split=20, ra
ndom_state=42)
model_5 = RandomForestRegressor(n_estimators=100, max_depth=7, random_stat
e=42)
## model selection and MAE function used as standard from kaggle ##

models = [model_1, model_2, model_3, model_4, model_5]

#function to calculate the MAE of each model to find the model of best fit
def score_model(model, X_t=train_X, X_v=val_X, y_t=train_y, y_v=val_y):
        model.fit(X_t, y_t)
        preds = model.predict(X_v)
        return mean_absolute_error(y_v, preds)

#for loop to run function on all model options
for i in range(0, len(models)):
    mae = score_model(models[i])
    print("Model",(i+1, mae))
```

```
#Best model = 3
model = model_3
Team_11_model = model
# Fit the model to the training data
Team_11_model.fit(X, y)

# Generate test predictions
Team_11_preds_test = Team_11_model.predict(X)
print(mean_absolute_error(y, Team_11_preds_test))

#function to run label groups through the model
def model(DF):

    model = model_3
    Team_11_model = model
# Fit the model to the training data
    Team_11_model.fit(X, y)

# Generate test predictions
    Team_11_preds_test = Team_11_model.predict(X)
    DF['MAE']= mean_absolute_error(y, Team_11_preds_test)
    #print(Team_11_preds_test)

    output = pd.DataFrame({'MAE':DF.MAE,'mfr':DF.mfr, 'Actual_Time': df.ac
tualleadtime,
                           'Expected_Time': Team_11_preds_test,
                           'Time_Difference':Team_11_preds_test - df.actua
lleadtime,
                           'label':DF.label})
    Results = pd.DataFrame(output)
    pd.DataFrame(Results)
    return Results

#function that will repeat the labels through the model
def repeater(arg):
    for i in range(0,336):
        W = label(arg, i)
        v = W.mean()
        pd.DataFrame(v)
        print(v)
```

```
#Function that brings everything together

def main():
    V= repeater(data)
    P= model(V)
    return P.to_csv('Final_Arrow_Model.csv', index = False)

#creating the data frame columns and information to represent in the csv f
ile

Final_csv = pd.DataFrame({'ID': data.label, 'Expected_Time':Team_11_preds_
test,'Actual_Time':df.actualleadtime,
                          'Time_Difference':df.actualleadtime - Team_11_pr
eds_test})

#Returning results to a csv
Final_csv.to_csv('result.csv', index = False)
```