In [47…

```
########################################
# importing packages
########################################
import matplotlib.pyplot as plt                       # data visualization
import pandas as pd                                   # data science essentials
from sklearn.model_selection import train_test_split  # train-test split
from sklearn.metrics import roc_auc_score             # auc score
from sklearn.model_selection import RandomizedSearchCV # hyperparameter tuning
from sklearn.metrics import make_scorer               # customizable scorer
from sklearn.metrics import confusion_matrix          # confusion matrix

#logistic
from sklearn.linear_model import LogisticRegression  # logistic regression
import statsmodels.formula.api as smf                # logistic regression
from sklearn.metrics import confusion_matrix         # confusion matrix
from sklearn.metrics import roc_auc_score            # auc score
from sklearn.neighbors import KNeighborsClassifier   # KNN for classification
from sklearn.neighbors import KNeighborsRegressor    # KNN for regression
from sklearn.preprocessing import StandardScaler     # standard scaler

# CART model packages
from sklearn.tree import DecisionTreeClassifier      # classification trees
from sklearn.tree import export_graphviz             # exports graphics
from six import StringIO                             # saves objects in memory
from IPython.display import Image                    # displays on frontend
import pydotplus

# Hypertuning packages
from sklearn.model_selection import RandomizedSearchCV     # hyperparameter tuning
from sklearn.metrics import make_scorer              # customizable scorer

#Ensemble Modeling packages
from sklearn.ensemble import RandomForestClassifier     # random forest
from sklearn.ensemble import GradientBoostingClassifier # gbm


########################################
# loading data and setting display options
########################################
# loading data
chef = pd.read_excel('./datasets/Apprentice_Chef_Dataset.xlsx')
```

In [47…

```
chef.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1946 entries, 0 to 1945
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   REVENUE                         1946 non-null   float64
 1   CROSS_SELL_SUCCESS              1946 non-null   int64
 2   NAME                            1946 non-null   object
 3   EMAIL                           1946 non-null   object
 4   FIRST_NAME                      1946 non-null   object
 5   FAMILY_NAME                     1899 non-null   object
 6   TOTAL_MEALS_ORDERED             1946 non-null   int64
 7   UNIQUE_MEALS_PURCH              1946 non-null   int64
 8   CONTACTS_W_CUSTOMER_SERVICE     1946 non-null   int64
 9   PRODUCT_CATEGORIES_VIEWED       1946 non-null   int64
 10  AVG_TIME_PER_SITE_VISIT         1946 non-null   float64
 11  MOBILE_NUMBER                   1946 non-null   int64
 12  CANCELLATIONS_BEFORE_NOON       1946 non-null   int64
 13  CANCELLATIONS_AFTER_NOON        1946 non-null   int64
```

```
 14  TASTES_AND_PREFERENCES       1946 non-null   int64
 15  PC_LOGINS                    1946 non-null   int64
 16  MOBILE_LOGINS                1946 non-null   int64
 17  WEEKLY_PLAN                  1946 non-null   int64
 18  EARLY_DELIVERIES             1946 non-null   int64
 19  LATE_DELIVERIES              1946 non-null   int64
 20  PACKAGE_LOCKER               1946 non-null   int64
 21  REFRIGERATED_LOCKER          1946 non-null   int64
 22  AVG_PREP_VID_TIME            1946 non-null   float64
 23  LARGEST_ORDER_SIZE           1946 non-null   int64
 24  MASTER_CLASSES_ATTENDED      1946 non-null   int64
 25  MEDIAN_MEAL_RATING           1946 non-null   int64
 26  AVG_CLICKS_PER_VISIT         1946 non-null   int64
 27  TOTAL_PHOTOS_VIEWED          1946 non-null   int64
dtypes: float64(3), int64(21), object(4)
memory usage: 425.8+ KB
```

In [48…

```
# pulling up data dictionary
chef_description = pd.read_excel('./datasets/Apprentice_Chef_Data_Dictionary.xlsx')


# displaying the data dictionary
chef_description
```

Out[480]:

	Feature	Data Type	Description
0	REVENUE	Float	Total revenue generated from each customer. No...
1	CROSS_SELL_SUCCESS	Integer	Success of the cross-sell promotion (1 = yes, ...
2	NAME	string	Full name of customer
3	EMAIL	string	Email of customer
4	FIRST_NAME	string	First name of customer
5	FAMILY_NAME	string	Last name of customer
6	TOTAL_MEALS_ORDERED	Integer	Total number of meals ordered by each customer...
7	UNIQUE_MEALS_PURCH	Integer	Number of unique meal sets ordered by each cus...
8	CONTACTS_W_CUSTOMER_SERVICE	Integer	Number of times each customer made contact wit...
9	PRODUCT_CATEGORIES_VIEWED	Integer	Total number of meal categories viewed (vegan,...
10	AVG_TIME_PER_SITE_VISIT	Float	Average time each customer spent per website o...
11	MOBILE_NUMBER	Integer	Indicates whether the registered phone number ...
12	CANCELLATIONS_BEFORE_NOON	Integer	Number of meals canceled BEFORE 12:00 PM (noon...
13	CANCELLATIONS_AFTER_NOON	Integer	Number of meals canceled AFTER 12:00 PM (noon)...
14	TASTES_AND_PREFERENCES	Integer	Customer specified their tastes and preference...
15	MOBILE_LOGINS	Integer	Total number of logins to the mobile platform
16	PC_LOGINS	Integer	Total number of logins to the website
17	WEEKLY_PLAN	Integer	Number of weeks customer subscribed to the wee...
18	EARLY_DELIVERIES	Integer	Total meal deliveries that arrived early
19	LATE_DELIVERIES	Integer	Total meal deliveries that arrived late
20	PACKAGE_LOCKER	Integer	Customer's residence/building has a package lo...
21	REFRIGERATED_LOCKER	Integer	Package locker has refrigerated compartments
22	AVG_PREP_VID_TIME	Float	Average time in seconds meal prep instruction ...

	Feature	Data Type	Description
23	LARGEST_ORDER_SIZE	Integer	MISLABELED: Despite the fact that this feature...
24	MASTER_CLASSES_ATTENDED	Integer	Number of times each customer attended a maste...
25	MEDIAN_MEAL_RATING	Integer	Median rating of meal sets by each customer
26	AVG_CLICKS_PER_VISIT	Float	Average number of clicks per site or mobile ap...
27	TOTAL_PHOTOS_VIEWED	Integer	Total number of clicks on photos across all we...

In [48…

```
# checking for missing values
chef.isnull().sum()
```

Out[481]:

```
REVENUE                      0
CROSS_SELL_SUCCESS           0
NAME                         0
EMAIL                        0
FIRST_NAME                   0
FAMILY_NAME                 47
TOTAL_MEALS_ORDERED          0
UNIQUE_MEALS_PURCH           0
CONTACTS_W_CUSTOMER_SERVICE  0
PRODUCT_CATEGORIES_VIEWED    0
AVG_TIME_PER_SITE_VISIT      0
MOBILE_NUMBER                0
CANCELLATIONS_BEFORE_NOON    0
CANCELLATIONS_AFTER_NOON     0
TASTES_AND_PREFERENCES       0
PC_LOGINS                    0
MOBILE_LOGINS                0
WEEKLY_PLAN                  0
EARLY_DELIVERIES             0
LATE_DELIVERIES              0
PACKAGE_LOCKER               0
REFRIGERATED_LOCKER          0
AVG_PREP_VID_TIME            0
LARGEST_ORDER_SIZE           0
MASTER_CLASSES_ATTENDED      0
MEDIAN_MEAL_RATING           0
AVG_CLICKS_PER_VISIT         0
TOTAL_PHOTOS_VIEWED          0
dtype: int64
```

In [48…

```
# splitting personal emails

# placeholder list
placeholder_lst = []

# looping over each email address
for index, col in chef.iterrows():

    # splitting email domain at '@'
    split_email = chef.loc[index, 'EMAIL'].split(sep = '@')

    # appending placeholder_lst with the results
    placeholder_lst.append(split_email)


# converting placeholder_lst into a DataFrame
email_df = pd.DataFrame(placeholder_lst)


# displaying the results
email_df
```

Out[482]:

	0	1
0	saathos	unitedhealth.com
1	alysanne.osgrey	ge.org
2	edwyd.fossoway	jnj.com
3	eleyna.westerling	ge.org
4	elyn.norridge	jnj.com
...	...	...
1941	obara.sand	yahoo.com
1942	quentyn.blackwood	yahoo.com
1943	rhonda.rowan	gmail.com
1944	turnip	yahoo.com
1945	tommard.heddle	merck.com

1946 rows × 2 columns

In [48…

```
#concatenating with original DataFrame

# renaming column to concatenate
email_df.columns = ['0' , 'personal_email_domain']


# concatenating personal_email_domain with friends DataFrame
chef = pd.concat([chef, email_df['personal_email_domain']],
                 axis = 1)


# printing value counts of personal_email_domain
chef.loc[: ,'personal_email_domain'].value_counts()
```

Out[483]:

```
gmail.com           303
protonmail.com      284
yahoo.com           274
msn.com              72
aol.com              69
passport.com         64
hotmail.com          63
live.com             62
me.com               59
amex.com             30
merck.com            28
mcdonalds.com        28
jnj.com              28
cocacola.com         28
nike.com             27
apple.com            27
ge.org               26
ibm.com              26
dupont.com           26
microsoft.com        25
chevron.com          25
unitedhealth.com     24
exxon.com            24
travelers.com        24
boeing.com           23
pg.com               22
```

```
verizon.com          22
mmm.com              22
caterpillar.com      22
walmart.com          21
disney.com           21
visa.com             20
pfizer.com           20
jpmorgan.com         19
cisco.com            18
unitedtech.com       18
goldmansacs.com      18
intel.com            17
homedepot.com        17
Name: personal_email_domain, dtype: int64
```

In [48…

```
personal_email_domains = ['@gmail.com','@yahoo.com','@protonmail.com']
junk_email_domains  = ['@me.com','@aol.com','@hotmail.com','@live.com','@msn.com','@passport.com']

# placeholder list
placeholder_lst = []


# looping to group observations by domain type
for domain in chef['personal_email_domain']:

    if '@' + domain in personal_email_domains:
        placeholder_lst.append('personal')


    elif '@' + domain in junk_email_domains:
        placeholder_lst.append('junk')

    else:
        placeholder_lst.append('professional')


# concatenating with original DataFrame
chef['domain_group'] = pd.Series(placeholder_lst)


# checking results
chef['domain_group'].value_counts()
```

Out[484]:

```
personal        861
professional    696
junk            389
Name: domain_group, dtype: int64
```

In [48…

```
def text_split_feature(col, df, sep=' ', new_col_name='number_of_names'):
    """
Splits values in a string Series (as part of a DataFrame) and sums the number
of resulting items. Automatically appends summed column to original DataFrame.

PARAMETERS
----------
col          : column to split
df           : DataFrame where column is located
sep          : string sequence to split by, default ' '
new_col_name : name of new column after summing split, default
               'number_of_names'
"""

    df[new_col_name] = 0
```

```
    for index, val in df.iterrows():
        df.loc[index, new_col_name] = len(df.loc[index, col].split(sep = ' '))
```

In [48…

```
# calling text_split_feature
text_split_feature(col = 'NAME',
                   df  = chef)


# checking results
chef['number_of_names'].value_counts().sort_index()
```

Out[486]:
```
1     591
2    1201
3      98
4       9
5      35
6      12
Name: number_of_names, dtype: int64
```

In [48…

```
# one hot encoding variables
skill       = pd.get_dummies(chef['MASTER_CLASSES_ATTENDED'])
domain_group = pd.get_dummies(chef['domain_group'])


# joining codings together
chef = chef.join([skill, domain_group])


# checking results
chef.columns
```

Out[487]:
```
Index([                    'REVENUE',          'CROSS_SELL_SUCCESS',
                              'NAME',                      'EMAIL',
                        'FIRST_NAME',                'FAMILY_NAME',
              'TOTAL_MEALS_ORDERED',          'UNIQUE_MEALS_PURCH',
       'CONTACTS_W_CUSTOMER_SERVICE',    'PRODUCT_CATEGORIES_VIEWED',
           'AVG_TIME_PER_SITE_VISIT',                 'MOBILE_NUMBER',
         'CANCELLATIONS_BEFORE_NOON',     'CANCELLATIONS_AFTER_NOON',
            'TASTES_AND_PREFERENCES',                   'PC_LOGINS',
                     'MOBILE_LOGINS',                'WEEKLY_PLAN',
                   'EARLY_DELIVERIES',             'LATE_DELIVERIES',
                    'PACKAGE_LOCKER',          'REFRIGERATED_LOCKER',
                 'AVG_PREP_VID_TIME',          'LARGEST_ORDER_SIZE',
         'MASTER_CLASSES_ATTENDED',          'MEDIAN_MEAL_RATING',
            'AVG_CLICKS_PER_VISIT',          'TOTAL_PHOTOS_VIEWED',
          'personal_email_domain',                 'domain_group',
                'number_of_names',                             0,
                                1,                             2,
                                3,                        'junk',
                       'personal',               'professional'],
      dtype='object')
```

In [48…

```
# dropping unfavorite features and categorical variables after they've been encoded
chef = chef.drop(labels = ['FIRST_NAME','FAMILY_NAME','NAME','EMAIL','MASTER_CLASSES_ATTENDED','domain_group','personal_email_domain'],
                 axis = 1)

# checking results
chef.columns
```

Out[488]:
```
Index([                    'REVENUE',          'CROSS_SELL_SUCCESS',
              'TOTAL_MEALS_ORDERED',          'UNIQUE_MEALS_PURCH',
       'CONTACTS_W_CUSTOMER_SERVICE',    'PRODUCT_CATEGORIES_VIEWED',
           'AVG_TIME_PER_SITE_VISIT',                 'MOBILE_NUMBER',
         'CANCELLATIONS_BEFORE_NOON',     'CANCELLATIONS_AFTER_NOON',
```

```
       'TASTES_AND_PREFERENCES',                 'PC_LOGINS',
              'MOBILE_LOGINS',               'WEEKLY_PLAN',
           'EARLY_DELIVERIES',           'LATE_DELIVERIES',
             'PACKAGE_LOCKER',        'REFRIGERATED_LOCKER',
          'AVG_PREP_VID_TIME',         'LARGEST_ORDER_SIZE',
         'MEDIAN_MEAL_RATING',       'AVG_CLICKS_PER_VISIT',
        'TOTAL_PHOTOS_VIEWED',           'number_of_names',
                            0,                           1,
                            2,                           3,
                       'junk',                    'personal',
                'professional'],
      dtype='object')
```

In [48…

```
# relabeling columns
chef.columns = [                  'REVENUE',          'CROSS_SELL_SUCCESS',
               'TOTAL_MEALS_ORDERED',          'UNIQUE_MEALS_PURCH',
       'CONTACTS_W_CUSTOMER_SERVICE',   'PRODUCT_CATEGORIES_VIEWED',
           'AVG_TIME_PER_SITE_VISIT',              'MOBILE_NUMBER',
         'CANCELLATIONS_BEFORE_NOON',    'CANCELLATIONS_AFTER_NOON',
            'TASTES_AND_PREFERENCES',                 'PC_LOGINS',
                    'MOBILE_LOGINS',               'WEEKLY_PLAN',
                 'EARLY_DELIVERIES',           'LATE_DELIVERIES',
                   'PACKAGE_LOCKER',        'REFRIGERATED_LOCKER',
                'AVG_PREP_VID_TIME',         'LARGEST_ORDER_SIZE',          'MEDIAN_MEAL_RATING',
             'AVG_CLICKS_PER_VISIT',  'TOTAL_PHOTOS_VIEWED',        'number_of_names',
               'skill_0','skill_1','skill_2','skill_3',
                           'junk',                    'personal',
                    'professional']


# checking results
chef.columns
```

Out[489]:

```
Index(['REVENUE', 'CROSS_SELL_SUCCESS', 'TOTAL_MEALS_ORDERED',
       'UNIQUE_MEALS_PURCH', 'CONTACTS_W_CUSTOMER_SERVICE',
       'PRODUCT_CATEGORIES_VIEWED', 'AVG_TIME_PER_SITE_VISIT', 'MOBILE_NUMBER',
       'CANCELLATIONS_BEFORE_NOON', 'CANCELLATIONS_AFTER_NOON',
       'TASTES_AND_PREFERENCES', 'PC_LOGINS', 'MOBILE_LOGINS', 'WEEKLY_PLAN',
       'EARLY_DELIVERIES', 'LATE_DELIVERIES', 'PACKAGE_LOCKER',
       'REFRIGERATED_LOCKER', 'AVG_PREP_VID_TIME', 'LARGEST_ORDER_SIZE',
       'MEDIAN_MEAL_RATING', 'AVG_CLICKS_PER_VISIT', 'TOTAL_PHOTOS_VIEWED',
       'number_of_names', 'skill_0', 'skill_1', 'skill_2', 'skill_3', 'junk',
       'personal', 'professional'],
      dtype='object')
```

In [49…

```
# User define function
```

In [49…

```
########################################
# optimal_neighbors
########################################
def optimal_neighbors(X_data,
                      y_data,
                      standardize = True,
                      pct_test=0.25,
                      seed=219,
                      response_type='reg',
                      max_neighbors=20,
                      show_viz=True):
    """
Exhaustively compute training and testing results for KNN across
[1, max_neighbors]. Outputs the maximum test score and (by default) a
visualization of the results.
PARAMETERS
----------
```

```
X_data        : explanatory variable data
y_data        : response variable
standardize   : whether or not to standardize the X data, default True
pct_test      : test size for training and validation from (0,1), default 0.25
seed          : random seed to be used in algorithm, default 219
response_type : type of neighbors algorithm to use, default 'reg'
    Use 'reg' for regression (KNeighborsRegressor)
    Use 'class' for classification (KNeighborsClassifier)
max_neighbors : maximum number of neighbors in exhaustive search, default 20
show_viz      : display or surpress k-neigbors visualization, default True
"""


    if standardize == True:
        # optionally standardizing X_data
        scaler             = StandardScaler()
        scaler.fit(X_data)
        X_scaled           = scaler.transform(X_data)
        X_scaled_df        = pd.DataFrame(X_scaled)
        X_data             = X_scaled_df



    # train-test split
    X_train, X_test, y_train, y_test = train_test_split(X_data,
                                                        y_data,
                                                        test_size = pct_test,
                                                        random_state = seed)


    # creating lists for training set accuracy and test set accuracy
    training_accuracy = []
    test_accuracy = []


    # setting neighbor range
    neighbors_settings = range(1, max_neighbors + 1)


    for n_neighbors in neighbors_settings:
        # building the model based on response variable type
        if response_type == 'reg':
            clf = KNeighborsRegressor(n_neighbors = n_neighbors)
            clf.fit(X_train, y_train)

        elif response_type == 'class':
            clf = KNeighborsClassifier(n_neighbors = n_neighbors)
            clf.fit(X_train, y_train)

        else:
            print("Error: response_type must be 'reg' or 'class'")


        # recording the training set accuracy
        training_accuracy.append(clf.score(X_train, y_train))

        # recording the generalization accuracy
        test_accuracy.append(clf.score(X_test, y_test))


    # optionally displaying visualization
    if show_viz == True:
        # plotting the visualization
        fig, ax = plt.subplots(figsize=(12,8))
        plt.plot(neighbors_settings, training_accuracy, label = "training accuracy")
        plt.plot(neighbors_settings, test_accuracy, label = "test accuracy")
```

```
        plt.ylabel("Accuracy")
        plt.xlabel("n_neighbors")
        plt.legend()
        plt.show()


    # returning optimal number of neighbors
    print(f"The optimal number of neighbors is: {test_accuracy.index(max(test_accuracy))+1}")
    return test_accuracy.index(max(test_accuracy))+1


########################################
# visual_cm
########################################
def visual_cm(true_y, pred_y, labels = None):
    """
Creates a visualization of a confusion matrix.

PARAMETERS
----------
true_y : true values for the response variable
pred_y : predicted values for the response variable
labels : , default None
    """
    # visualizing the confusion matrix

    # setting labels
    lbls = labels


    # declaring a confusion matrix object
    cm = confusion_matrix(y_true = true_y,
                          y_pred = pred_y)


    # heatmap
    sns.heatmap(cm,
                annot       = True,
                xticklabels = lbls,
                yticklabels = lbls,
                cmap        = 'Blues',
                fmt         = 'g')


    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix of the Classifier')
    plt.show()
```

In [49…

```
df_corr = chef.corr().round(2)

df_corr['CROSS_SELL_SUCCESS'].sort_values(ascending = False)
```

Out[492]:

```
CROSS_SELL_SUCCESS             1.00
professional                   0.19
number_of_names                0.16
CANCELLATIONS_BEFORE_NOON      0.16
MOBILE_NUMBER                  0.10
TASTES_AND_PREFERENCES         0.08
REFRIGERATED_LOCKER            0.07
skill_1                        0.05
CONTACTS_W_CUSTOMER_SERVICE    0.04
PC_LOGINS                      0.04
PACKAGE_LOCKER                 0.04
```

```
personal                      0.04
MEDIAN_MEAL_RATING            0.03
AVG_PREP_VID_TIME             0.03
LARGEST_ORDER_SIZE            0.02
EARLY_DELIVERIES              0.02
AVG_TIME_PER_SITE_VISIT       0.01
TOTAL_MEALS_ORDERED           0.01
LATE_DELIVERIES               0.01
TOTAL_PHOTOS_VIEWED           0.01
skill_2                       0.01
PRODUCT_CATEGORIES_VIEWED     0.00
UNIQUE_MEALS_PURCH            0.00
REVENUE                       0.00
WEEKLY_PLAN                  -0.01
AVG_CLICKS_PER_VISIT         -0.04
skill_3                      -0.04
MOBILE_LOGINS                -0.05
CANCELLATIONS_AFTER_NOON     -0.05
skill_0                      -0.05
junk                         -0.28
Name: CROSS_SELL_SUCCESS, dtype: float64
```

In [49…

```
# declaring explanatory variables
chef_data = chef.drop('CROSS_SELL_SUCCESS', axis = 1)


# declaring response variable
chef_target = chef.loc[ : , 'CROSS_SELL_SUCCESS']
```

In [49…

```
chef.describe()
```

Out[494]:

	REVENUE	CROSS_SELL_SUCCESS	TOTAL_MEALS_ORDERED	UNIQUE_MEALS_PURCH	CONTACTS_W_CUSTOMER_SERVICE	PRODUCT_CATEGORIES_VIEWED	AVG_TIME_PER_SITE_VISIT	MOBILE_NUMBER	CANCEL
count	1946.000000	1946.000000	1946.000000	1946.000000	1946.000000	1946.000000	1946.000000	1946.000000	
mean	2107.292652	0.678828	74.634121	4.904933	6.983556	5.383864	99.604651	0.877698	
std	1138.290709	0.467047	55.309782	2.502175	2.281193	3.044001	62.341756	0.327719	
min	131.000000	0.000000	11.000000	1.000000	1.000000	1.000000	10.330000	0.000000	
25%	1350.000000	0.000000	39.000000	3.000000	5.000000	3.000000	72.000000	1.000000	
50%	1740.000000	1.000000	60.000000	5.000000	7.000000	5.000000	94.160000	1.000000	
75%	2670.000000	1.000000	95.000000	7.000000	8.000000	8.000000	117.287500	1.000000	
max	8793.750000	1.000000	493.000000	19.000000	18.000000	10.000000	1645.600000	1.000000	

8 rows × 31 columns

In [49…

```
# train-test split with stratification
X_train, X_test, y_train, y_test = train_test_split(
            chef_data,
            chef_target,
            test_size    = 0.25,
            random_state = 219,
            stratify     = chef_target)


# merging training data for statsmodels
chef_train = pd.concat([X_train, y_train], axis = 1)
```

In [49…
```
print(f"""

Response Variable Proportions (Training Set)
--------------------------------------------
{y_train.value_counts(normalize = True).round(decimals = 2)}



Response Variable Proportions (Testing Set)
--------------------------------------------
{y_test.value_counts(normalize = True).round(decimals = 2)}
""")
```

```
Response Variable Proportions (Training Set)
--------------------------------------------
1    0.68
0    0.32
Name: CROSS_SELL_SUCCESS, dtype: float64



Response Variable Proportions (Testing Set)
--------------------------------------------
1    0.68
0    0.32
Name: CROSS_SELL_SUCCESS, dtype: float64
```

In [49…
```
for val in chef_data:
    print(f" {val} + ")
```

```
REVENUE +
TOTAL_MEALS_ORDERED +
UNIQUE_MEALS_PURCH +
CONTACTS_W_CUSTOMER_SERVICE +
PRODUCT_CATEGORIES_VIEWED +
AVG_TIME_PER_SITE_VISIT +
MOBILE_NUMBER +
CANCELLATIONS_BEFORE_NOON +
CANCELLATIONS_AFTER_NOON +
TASTES_AND_PREFERENCES +
PC_LOGINS +
MOBILE_LOGINS +
WEEKLY_PLAN +
EARLY_DELIVERIES +
LATE_DELIVERIES +
PACKAGE_LOCKER +
REFRIGERATED_LOCKER +
AVG_PREP_VID_TIME +
LARGEST_ORDER_SIZE +
MEDIAN_MEAL_RATING +
AVG_CLICKS_PER_VISIT +
TOTAL_PHOTOS_VIEWED +
number_of_names +
skill_0 +
skill_1 +
skill_2 +
skill_3 +
junk +
personal +
professional +
```

In [49…
```
# instantiating a logistic regression model object
logistic_full = smf.logit(formula = """  CROSS_SELL_SUCCESS ~
REVENUE +
 TOTAL_MEALS_ORDERED +
```

```
UNIQUE_MEALS_PURCH +
CONTACTS_W_CUSTOMER_SERVICE +
PRODUCT_CATEGORIES_VIEWED +
AVG_TIME_PER_SITE_VISIT +
MOBILE_NUMBER +
CANCELLATIONS_BEFORE_NOON +
CANCELLATIONS_AFTER_NOON +
TASTES_AND_PREFERENCES +
PC_LOGINS +
MOBILE_LOGINS +
WEEKLY_PLAN +
EARLY_DELIVERIES +
LATE_DELIVERIES +
PACKAGE_LOCKER +
REFRIGERATED_LOCKER +
AVG_PREP_VID_TIME +
LARGEST_ORDER_SIZE +
MEDIAN_MEAL_RATING +
AVG_CLICKS_PER_VISIT +
number_of_names +
skill_0 +
skill_1 +
skill_2 +
skill_3 +
junk +
personal +
professional""",
                                    data    = chef_train)


# fitting the model object
results_full = logistic_full.fit()


# checking the results SUMMARY
results_full.summary()
```

```
Optimization terminated successfully.
        Current function value: 0.533805
        Iterations 6
```

Out[498]:

Logit Regression Results

Dep. Variable:	CROSS_SELL_SUCCESS	**No. Observations:**	1459
Model:	Logit	**Df Residuals:**	1431
Method:	MLE	**Df Model:**	27
Date:	Tue, 26 Jan 2021	**Pseudo R-squ.:**	0.1499
Time:	23:28:26	**Log-Likelihood:**	-778.82
converged:	True	**LL-Null:**	-916.19
Covariance Type:	nonrobust	**LLR p-value:**	7.443e-43

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-2.2156	nan	nan	nan	nan	nan
REVENUE	-0.0002	8.89e-05	-2.394	0.017	-0.000	-3.86e-05
TOTAL_MEALS_ORDERED	-0.0005	0.001	-0.323	0.747	-0.003	0.002
UNIQUE_MEALS_PURCH	-0.0113	0.026	-0.431	0.666	-0.063	0.040
CONTACTS_W_CUSTOMER_SERVICE	0.0507	0.028	1.796	0.073	-0.005	0.106
PRODUCT_CATEGORIES_VIEWED	-0.0175	0.021	-0.849	0.396	-0.058	0.023

AVG_TIME_PER_SITE_VISIT	0.0003	0.001	0.301	0.763	-0.002	0.002
MOBILE_NUMBER	0.8987	0.179	5.023	0.000	0.548	1.249
CANCELLATIONS_BEFORE_NOON	0.2795	0.047	5.947	0.000	0.187	0.372
CANCELLATIONS_AFTER_NOON	-0.2471	0.144	-1.721	0.085	-0.528	0.034
TASTES_AND_PREFERENCES	0.3622	0.137	2.641	0.008	0.093	0.631
PC_LOGINS	0.2386	0.109	2.195	0.028	0.026	0.452
MOBILE_LOGINS	-0.1849	0.119	-1.552	0.121	-0.419	0.049
WEEKLY_PLAN	0.0056	0.005	1.179	0.238	-0.004	0.015
EARLY_DELIVERIES	0.0659	0.028	2.353	0.019	0.011	0.121
LATE_DELIVERIES	0.0163	0.023	0.713	0.476	-0.028	0.061
PACKAGE_LOCKER	0.0328	0.149	0.221	0.825	-0.259	0.325
REFRIGERATED_LOCKER	0.4995	0.239	2.094	0.036	0.032	0.967
AVG_PREP_VID_TIME	0.0050	0.003	1.904	0.057	-0.000	0.010
LARGEST_ORDER_SIZE	-0.0773	0.070	-1.106	0.269	-0.214	0.060
MEDIAN_MEAL_RATING	0.1330	0.170	0.780	0.435	-0.201	0.467
AVG_CLICKS_PER_VISIT	-0.0185	0.052	-0.358	0.721	-0.120	0.083
number_of_names	0.5530	0.095	5.835	0.000	0.367	0.739
skill_0	-0.4989	nan	nan	nan	nan	nan
skill_1	-0.2055	nan	nan	nan	nan	nan
skill_2	-0.1875	nan	nan	nan	nan	nan
skill_3	-1.3238	nan	nan	nan	nan	nan
junk	-1.8436	nan	nan	nan	nan	nan
personal	-0.4943	nan	nan	nan	nan	nan
professional	0.1223	nan	nan	nan	nan	nan

In [49…

```
#Develop a model where all features are significant based on their p-values
```

In [50…

```
# instantiating a logistic regression model object
logit_sig = smf.logit(formula = """ CROSS_SELL_SUCCESS ~
MOBILE_NUMBER+
                                    REVENUE +
                                    CANCELLATIONS_BEFORE_NOON +
                                    TASTES_AND_PREFERENCES +
                                    PC_LOGINS +
                                    EARLY_DELIVERIES+
                                    REFRIGERATED_LOCKER+
                                    number_of_names
                                    """,
                                    data = chef_train)


# fitting the model object
logit_sig = logit_sig.fit()
```

```
# checking the results SUMMARY
logit_sig.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.590401
         Iterations 6
```

Out[500]:

Logit Regression Results

Dep. Variable:	CROSS_SELL_SUCCESS	**No. Observations:**	1459
Model:	Logit	**Df Residuals:**	1450
Method:	MLE	**Df Model:**	8
Date:	Tue, 26 Jan 2021	**Pseudo R-squ.:**	0.05981
Time:	23:28:27	**Log-Likelihood:**	-861.40
converged:	True	**LL-Null:**	-916.19
Covariance Type:	nonrobust	**LLR p-value:**	4.609e-20

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-2.5476	0.618	-4.120	0.000	-3.760	-1.336
MOBILE_NUMBER	0.7225	0.168	4.308	0.000	0.394	1.051
REVENUE	3.549e-05	5.25e-05	0.676	0.499	-6.75e-05	0.000
CANCELLATIONS_BEFORE_NOON	0.2624	0.045	5.842	0.000	0.174	0.350
TASTES_AND_PREFERENCES	0.3586	0.128	2.812	0.005	0.109	0.609
PC_LOGINS	0.1893	0.101	1.877	0.060	-0.008	0.387
EARLY_DELIVERIES	0.0506	0.026	1.955	0.051	-0.000	0.101
REFRIGERATED_LOCKER	0.4659	0.199	2.340	0.019	0.076	0.856
number_of_names	0.4743	0.088	5.364	0.000	0.301	0.648

In [52…

```
# explanatory sets from last session

# creating a dictionary to store candidate models

candidate_dict = {

 # full model
       'logit_full'   :        ['TOTAL_MEALS_ORDERED',          'UNIQUE_MEALS_PURCH',
      'CONTACTS_W_CUSTOMER_SERVICE',    'PRODUCT_CATEGORIES_VIEWED',
          'AVG_TIME_PER_SITE_VISIT',              'MOBILE_NUMBER',
        'CANCELLATIONS_BEFORE_NOON',    'CANCELLATIONS_AFTER_NOON',
           'TASTES_AND_PREFERENCES',                   'PC_LOGINS',
                    'MOBILE_LOGINS',                'WEEKLY_PLAN',
                 'EARLY_DELIVERIES',             'LATE_DELIVERIES',
                   'PACKAGE_LOCKER',         'REFRIGERATED_LOCKER',
                'AVG_PREP_VID_TIME',          'LARGEST_ORDER_SIZE',          'MEDIAN_MEAL_RATING',
             'AVG_CLICKS_PER_VISIT',          'number_of_names',
               'skill_0','skill_1','skill_2','skill_3',
                           'junk',                     'personal',
                  'professional'],


 # significant variables only (set 1)
 'logit_sig'     : ['MOBILE_NUMBER', 'CANCELLATIONS_BEFORE_NOON' ,
                                    'TASTES_AND_PREFERENCES' ,
                                    'number_of_names'
```

```
                                    ],


 # significant variables only (set 2)
 'logit_sig_2'  :  ['REVENUE' ,'MOBILE_NUMBER', 'CANCELLATIONS_BEFORE_NOON' ,
                                    'TASTES_AND_PREFERENCES' ,
                                    'PC_LOGINS' ,
                                    'EARLY_DELIVERIES',
                                    'REFRIGERATED_LOCKER',
                                    'number_of_names','skill_0','skill_1','skill_2','skill_3',  'junk',                    'personal',
                       'professional'
                                    ]

}
```

In [52…

```
# printing candidate variable sets
print(f"""
/--------------------------\\
|Explanatory Variable Sets |
\\--------------------------/

Full Model:
-----------
{candidate_dict['logit_full']}


First Significant p-value Model:
--------------------------------
{candidate_dict['logit_sig']}


Second Significant p-value Model:
---------------------------------
{candidate_dict['logit_sig_2']}
""")
```

```
/--------------------------\
|Explanatory Variable Sets |
\--------------------------/

Full Model:
-----------
['TOTAL_MEALS_ORDERED', 'UNIQUE_MEALS_PURCH', 'CONTACTS_W_CUSTOMER_SERVICE', 'PRODUCT_CATEGORIES_VIEWED', 'AVG_TIME_PER_SITE_VISIT', 'MOBILE_NUMBER', 'CANCELLATIONS_BEFORE_NOON', 'CANC
ELLATIONS_AFTER_NOON', 'TASTES_AND_PREFERENCES', 'PC_LOGINS', 'MOBILE_LOGINS', 'WEEKLY_PLAN', 'EARLY_DELIVERIES', 'LATE_DELIVERIES', 'PACKAGE_LOCKER', 'REFRIGERATED_LOCKER', 'AVG_PREP_
VID_TIME', 'LARGEST_ORDER_SIZE', 'MEDIAN_MEAL_RATING', 'AVG_CLICKS_PER_VISIT', 'number_of_names', 'skill_0', 'skill_1', 'skill_2', 'skill_3', 'junk', 'personal', 'professional']


First Significant p-value Model:
--------------------------------
['MOBILE_NUMBER', 'CANCELLATIONS_BEFORE_NOON', 'TASTES_AND_PREFERENCES', 'number_of_names']


Second Significant p-value Model:
---------------------------------
['REVENUE', 'MOBILE_NUMBER', 'CANCELLATIONS_BEFORE_NOON', 'TASTES_AND_PREFERENCES', 'PC_LOGINS', 'EARLY_DELIVERIES', 'REFRIGERATED_LOCKER', 'number_of_names', 'skill_0', 'skill_1', 'sk
ill_2', 'skill_3', 'junk', 'personal', 'professional']
```

In [53…

```
chef
```

Out[531]:

	REVENUE	CROSS_SELL_SUCCESS	TOTAL_MEALS_ORDERED	UNIQUE_MEALS_PURCH	CONTACTS_W_CUSTOMER_SERVICE	PRODUCT_CATEGORIES_VIEWED	AVG_TIME_PER_SITE_VISIT	MOBILE_NUMBER	CANCELLAT
0	393.0	1	14	6	12	10	48.00	1	

	REVENUE	CROSS_SELL_SUCCESS	TOTAL_MEALS_ORDERED	UNIQUE_MEALS_PURCH	CONTACTS_W_CUSTOMER_SERVICE	PRODUCT_CATEGORIES_VIEWED	AVG_TIME_PER_SITE_VISIT	MOBILE_NUMBER	CANCELLAT
1	1365.0	1	87	3	8	8	40.35	1	
2	800.0	1	15	7	11	5	19.77	1	
3	600.0	1	13	6	11	5	90.00	1	
4	1490.0	1	47	8	6	10	40.38	1	
...	...	...	...	...	...	...	...	...	
1941	3450.0	0	87	8	8	7	108.90	1	
1942	5829.0	0	244	4	7	2	133.91	1	
1943	1900.0	0	57	2	8	4	102.71	1	
1944	1600.0	0	74	3	10	10	638.87	0	
1945	2050.0	1	188	4	9	5	71.45	1	

1946 rows × 31 columns

In [53…

```
chef_data   =  chef.loc[ : , candidate_dict['logit_sig']]
chef_data
```

Out[532]:

	MOBILE_NUMBER	CANCELLATIONS_BEFORE_NOON	TASTES_AND_PREFERENCES	number_of_names
0	1	3	1	1
1	1	0	1	2
2	1	3	1	2
3	1	2	1	2
4	1	0	0	2
...	...	...	...	...
1941	1	0	1	2
1942	1	1	1	2
1943	1	2	1	2
1944	0	0	1	1
1945	1	4	1	2

1946 rows × 4 columns

In [53…

```
chef_target =  chef.loc[ : , 'CROSS_SELL_SUCCESS']
chef_target
```

Out[533]:

```
0       1
1       1
2       1
3       1
4       1
       ..
1941    0
1942    0
1943    0
```

```
1944    0
1945    1
Name: CROSS_SELL_SUCCESS, Length: 1946, dtype: int64
```

In [53…

```
# train/test split with the full model
chef_data   =  chef.loc[ : , candidate_dict['logit_sig_2']]
chef_target =  chef.loc[ : , 'CROSS_SELL_SUCCESS']

# this is the exact code we were using before

X_train, X_test, y_train, y_test = train_test_split(
            chef_data,
            chef_target,
            random_state = 219,
            test_size    = 0.25,
            stratify     = chef_target)


# INSTANTIATING a logistic regression model
logreg = LogisticRegression(solver = 'lbfgs',
                            C = 1,
                            random_state = 219)


# FITTING the training data
logreg_fit = logreg.fit(X_train, y_train)


# PREDICTING based on the testing set
logreg_pred = logreg_fit.predict(X_test)


# SCORING the results
print('LogReg Training ACCURACY:', logreg_fit.score(X_train, y_train).round(4))
print('LogReg Testing  ACCURACY:', logreg_fit.score(X_test, y_test).round(4))

# saving scoring data for future use
logreg_train_score = logreg_fit.score(X_train, y_train).round(4) # accuracy
logreg_test_score  = logreg_fit.score(X_test, y_test).round(4)   # accuracy


# displaying and saving the gap between training and testing
print('LogReg Train-Test Gap   :', abs(logreg_train_score - logreg_test_score).round(4))
logreg_test_gap = abs(logreg_train_score - logreg_test_score).round(4)
```

```
LogReg Training ACCURACY: 0.7231
LogReg Testing  ACCURACY: 0.7207
LogReg Train-Test Gap   : 0.0024
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [53…

```
# unpacking the confusion matrix
logreg_tn, \
logreg_fp, \
logreg_fn, \
logreg_tp = confusion_matrix(y_true = y_test, y_pred = logreg_pred).ravel()


# printing each result one-by-one
```

```
print(f"""
True Negatives : {logreg_tn}
False Positives: {logreg_fp}
False Negatives: {logreg_fn}
True Positives : {logreg_tp}
""")
```

```
True Negatives : 49
False Positives: 107
False Negatives: 29
True Positives : 302
```

In [53…

```
# area under the roc curve (auc)
print(roc_auc_score(y_true  = y_test,
                    y_score = logreg_pred).round(decimals = 4))


# saving AUC score for future use
logreg_auc_score = roc_auc_score(y_true  = y_test,
                                 y_score = logreg_pred).round(decimals = 4)
```

```
0.6132
```

In [53…

```
# zipping each feature name to its coefficient
logreg_model_values = zip(chef[candidate_dict['logit_sig']].columns,
                          logreg_fit.coef_.ravel().round(decimals = 2))


# setting up a placeholder list to store model features
logreg_model_lst = [('intercept', logreg_fit.intercept_[0].round(decimals = 2))]


# printing out each feature-coefficient pair one by one
for val in logreg_model_values:
    logreg_model_lst.append(val)


# checking the results
for pair in logreg_model_lst:
    print(pair)
```

```
('intercept', -0.16)
('MOBILE_NUMBER', -0.0)
('CANCELLATIONS_BEFORE_NOON', 0.48)
('TASTES_AND_PREFERENCES', 0.27)
('number_of_names', 0.27)
```

In [53…

```
# Classification Trees (CART Models)
```

In [53…

```
########################################
# display_tree
########################################
def display_tree(tree, feature_df, height = 500, width = 800):
    """
    PARAMETERS
    ----------
    tree       : fitted tree model object
        fitted CART model to visualized
    feature_df : DataFrame
        DataFrame of explanatory features (used to generate labels)
    height     : int, default 500
        height in pixels to which to constrain image in html
```

```
    width      : int, default 800
        width in pixels to which to constrain image in html
    """

    # visualizing the tree
    dot_data = StringIO()


    # exporting tree to graphviz
    export_graphviz(decision_tree      = tree,
                    out_file           = dot_data,
                    filled             = True,
                    rounded            = True,
                    special_characters = True,
                    feature_names      = feature_df.columns)


    # declaring a graph object
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())


    # creating image
    img = Image(graph.create_png(),
                height = height,
                width  = width)

    return img

########################################
# plot_feature_importances
########################################
def plot_feature_importances(model, train, export = False):
    """
    Plots the importance of features from a CART model.

    PARAMETERS
    ----------
    model  : CART model
    train  : explanatory variable training data
    export : whether or not to export as a .png image, default False
    """

    # declaring the number
    n_features = X_train.shape[1]

    # setting plot window
    fig, ax = plt.subplots(figsize=(12,9))

    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(pd.np.arange(n_features), train.columns)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")

    if export == True:
        plt.savefig('Tree_Leaf_50_Feature_Importance.png')
```

In [54…

```
# INSTANTIATING a classification tree object
full_tree = DecisionTreeClassifier()


# FITTING the training data
full_tree_fit = full_tree.fit(X_train, y_train)
```

```
# PREDICTING on new data
full_tree_pred = full_tree_fit.predict(X_test)


# SCORING the model
print('Full Tree Training ACCURACY:', full_tree_fit.score(X_train,
                                                          y_train).round(4))

print('Full Tree Testing ACCURACY :', full_tree_fit.score(X_test,
                                                          y_test).round(4))

print('Full Tree AUC Score:', roc_auc_score(y_true  = y_test,
                                            y_score = full_tree_pred).round(4))


# saving scoring data for future use
full_tree_train_score = full_tree_fit.score(X_train, y_train).round(4) # accuracy
full_tree_test_score  = full_tree_fit.score(X_test, y_test).round(4)   # accuracy


# saving AUC
full_tree_auc_score   = roc_auc_score(y_true  = y_test,
                                      y_score = full_tree_pred).round(4) # auc
```

```
Full Tree Training ACCURACY: 1.0
Full Tree Testing ACCURACY : 0.6735
Full Tree AUC Score: 0.6192
```

In [54…

```
# unpacking the confusion matrix
full_tree_tn, \
full_tree_fp, \
full_tree_fn, \
full_tree_tp = confusion_matrix(y_true = y_test, y_pred = full_tree_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {full_tree_tn}
False Positives: {full_tree_fp}
False Negatives: {full_tree_fn}
True Positives : {full_tree_tp}
""")
```

```
True Negatives : 73
False Positives: 83
False Negatives: 76
True Positives : 255
```

In [54…

```
# Puned tree
```

In [54…

```
# INSTANTIATING a classification tree object
pruned_tree = DecisionTreeClassifier(max_depth = 4,
                                     min_samples_leaf = 25,
                                     random_state = 219)


# FITTING the training data
pruned_tree_fit  = pruned_tree.fit(X_train, y_train)


# PREDICTING on new data
pruned_tree_pred = pruned_tree_fit.predict(X_test)
```

```
# SCORING the model
print('Training ACCURACY:', pruned_tree_fit.score(X_train, y_train).round(4))
print('Testing  ACCURACY:', pruned_tree_fit.score(X_test, y_test).round(4))
print('AUC Score        :', roc_auc_score(y_true  = y_test,
                                          y_score = pruned_tree_pred).round(4))


# saving scoring data for future use
pruned_tree_train_score = pruned_tree_fit.score(X_train, y_train).round(4) # accuracy
pruned_tree_test_score  = pruned_tree_fit.score(X_test, y_test).round(4)   # accuracy


# saving auc score
pruned_tree_auc_score   = roc_auc_score(y_true  = y_test,
                                        y_score = pruned_tree_pred).round(4) # auc
```

```
Training ACCURACY: 0.7512
Testing  ACCURACY: 0.7803
AUC Score        : 0.713
```

In [54…

```
# unpacking the confusion matrix
pruned_tree_tn, \
pruned_tree_fp, \
pruned_tree_fn, \
pruned_tree_tp = confusion_matrix(y_true = y_test, y_pred = pruned_tree_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {pruned_tree_tn}
False Positives: {pruned_tree_fp}
False Negatives: {pruned_tree_fn}
True Positives : {pruned_tree_tp}
""")
```

```
True Negatives : 82
False Positives: 74
False Negatives: 33
True Positives : 298
```

In [54…

```
# comparing results
print(f"""
Model         AUC Score      TN, FP, FN, TP
-----         ---------      --------------
Logistic      {logreg_auc_score}         {logreg_tn, logreg_fp, logreg_fn, logreg_tp}
Full Tree     {full_tree_auc_score}         {full_tree_tn, full_tree_fp, full_tree_fn, full_tree_tp}
Pruned Tree   {pruned_tree_auc_score}         {pruned_tree_tn, pruned_tree_fp, pruned_tree_fn, pruned_tree_tp}
""")


# creating a dictionary for model results
model_performance = {

    'Model Name'    : ['Logistic', 'Full Tree', 'Pruned Tree'],

    'AUC Score' : [logreg_auc_score, full_tree_auc_score, pruned_tree_auc_score],

    'Training Accuracy' : [logreg_train_score, full_tree_train_score,
                           pruned_tree_train_score],

    'Testing Accuracy'  : [logreg_test_score, full_tree_test_score,
                           pruned_tree_test_score],
```

```
    'Confusion Matrix'   : [(logreg_tn, logreg_fp, logreg_fn, logreg_tp),
                            (full_tree_tn, full_tree_fp, full_tree_fn, full_tree_tp),
                            (pruned_tree_tn, pruned_tree_fp, pruned_tree_fn, pruned_tree_tp)]}


# converting model_performance into a DataFrame
model_performance = pd.DataFrame(model_performance)


# sending model results to Excel
model_performance.to_excel('./model_results/classification_model_performance.xlsx',
                           index = False)
```

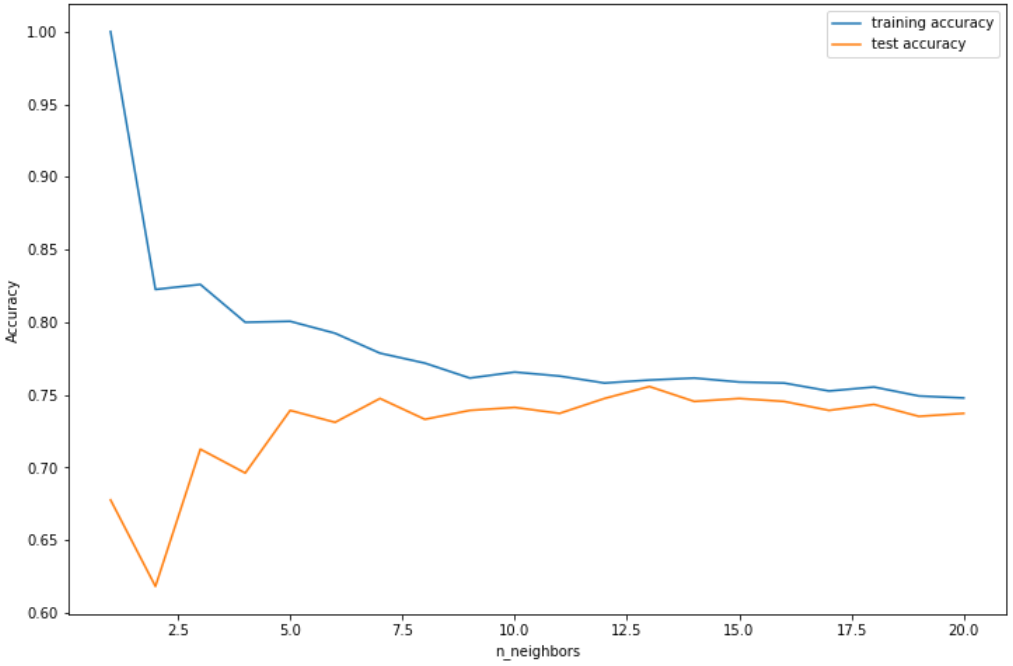
```
Model         AUC Score      TN, FP, FN, TP
-----         ---------      --------------
Logistic      0.6132          (49, 107, 29, 302)
Full Tree     0.6192          (73, 83, 76, 255)
Pruned Tree   0.713          (82, 74, 33, 298)
```

In [54…

```
#Classification Modeling with KNN
```

In [54…

```
# determining the optimal number of neighbors
opt_neighbors = optimal_neighbors(X_data       = chef_data,
                                  y_data       = chef_target,
                                  response_type = 'class')
```




```
The optimal number of neighbors is: 13
```

In [56…

```
# INSTANTIATING StandardScaler()
scaler = StandardScaler()
```

```
# FITTING the data
scaler.fit(chef_data)


# TRANSFORMING the data
X_scaled     = scaler.transform(chef_data)


# converting to a DataFrame
X_scaled_df  = pd.DataFrame(X_scaled)


# train-test split with the scaled data
X_train_scaled, X_test_scaled, y_train_scaled, y_test_scaled = train_test_split(
            X_scaled_df,
            chef_target,
            random_state = 219,
            test_size = 0.25,
            stratify = chef_target)


# INSTANTIATING a KNN classification model with optimal neighbors
knn_opt = KNeighborsClassifier(n_neighbors = opt_neighbors)


# FITTING the training data
knn_fit = knn_opt.fit(X_train_scaled, y_train_scaled)


# PREDICTING based on the testing set
knn_pred = knn_fit.predict(X_test_scaled)


# SCORING the results
print('Training ACCURACY:', knn_fit.score(X_train_scaled, y_train_scaled).round(4))
print('Testing  ACCURACY:', knn_fit.score(X_test_scaled, y_test_scaled).round(4))
print('AUC Score        :', roc_auc_score(y_true  = y_test,
                                          y_score = knn_pred).round(4))


# saving scoring data
knn_train_score = knn_fit.score(X_train_scaled, y_train_scaled).round(4)
knn_test_score  = knn_fit.score(X_test_scaled, y_test_scaled).round(4)


# saving AUC score
knn_auc_score   = roc_auc_score(y_true  = y_test,
                                          y_score = knn_pred).round(4)
```

```
Training ACCURACY: 0.7375
Testing  ACCURACY: 0.7064
AUC Score        : 0.5942
```

In [56…

```
# unpacking the confusion matrix
knn_tree_tn, \
knn_tree_fp, \
knn_tree_fn, \
knn_tree_tp = confusion_matrix(y_true = y_test, y_pred = knn_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {pruned_tree_tn}
False Positives: {pruned_tree_fp}
```

```
False Negatives: {pruned_tree_fn}
True Positives : {pruned_tree_tp}
""")
```

```
True Negatives : 82
False Positives: 74
False Negatives: 33
True Positives : 298
```

In [ ]:

In [57…

```
# Hyperparameter Tuning
```

In [59…

```
# train/test split with the logit_sig variables
chef_data   =  chef.loc[ : , candidate_dict['logit_sig_2']]
chef_target =  chef.loc[ : , 'CROSS_SELL_SUCCESS']


# train/test split
X_train, X_test, y_train, y_test = train_test_split(
            chef_data,
            chef_target,
            random_state = 219,
            test_size    = 0.25,
            stratify     = chef_target)
```

In [59…

```
########################################
# RandomizedSearchCV
########################################

# declaring a hyperparameter space
C_space          = pd.np.arange(0.1, 5.0, 0.1)
warm_start_space = [True, False]
solver_space     = ['newton-cg', 'sag', 'lbfgs']


# creating a hyperparameter grid
param_grid = {'C'          : C_space,
              'warm_start' : warm_start_space,
              'solver'     : solver_space}


# INSTANTIATING the model object without hyperparameters
lr_tuned = LogisticRegression(random_state = 219,
                              max_iter     = 1000)


# GridSearchCV object
lr_tuned_cv = RandomizedSearchCV(estimator           = lr_tuned,   # the model object
                                 param_distributions = param_grid, # parameters to tune
                                 cv                  = 3,          # how many folds in cross-validation
                                 n_iter              = 250,        # number of combinations of hyperparameters to try
                                 random_state        = 219,        # starting point for random sequence
                                 scoring = make_scorer(
                                           roc_auc_score,
                                           needs_threshold = False)) # scoring criteria (AUC)


# FITTING to the FULL DATASET (due to cross-validation)
lr_tuned_cv.fit(chef_data, chef_target)
```

```python
# PREDICT step is not needed


# printing the optimal parameters and best score
print("Tuned Parameters  :", lr_tuned_cv.best_params_)
print("Tuned CV AUC      :", lr_tuned_cv.best_score_.round(4))
```

```
<ipython-input-599-8b2c492ff40a>:6: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  C_space          = pd.np.arange(0.1, 5.0, 0.1)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/optimize.py:204: UserWarning: Line Search failed
  warnings.warn('Line Search failed')
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/optimize.py:204: UserWarning: Line Search failed
  warnings.warn('Line Search failed')
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/optimize.py:204: UserWarning: Line Search failed
  warnings.warn('Line Search failed')
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:437: LineSearchWarning: Rounding errors prevent the line search from converging
  warn(msg, LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/optimize.py:204: UserWarning: Line Search failed
  warnings.warn('Line Search failed')
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

```
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:477: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:437: LineSearchWarning: Rounding errors prevent the line search from converging
  warn(msg, LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/scipy/optimize/linesearch.py:327: LineSearchWarning: The line search algorithm did not converge
  warn('The line search algorithm did not converge', LineSearchWarning)
/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/optimize.py:204: UserWarning: Line Search failed
  warnings.warn('Line Search failed')
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
Tuned Parameters  : {'warm_start': True, 'solver': 'newton-cg', 'C': 0.6}
Tuned CV AUC      : 0.6549
/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_sag.py:329: ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
  warnings.warn("The max_iter was reached which means "
```

In [60…

```
# checking the results of RandomizedSearch CV
lr_tuned_cv.cv_results_
```

Out[600]:

```
{'mean_fit_time': array([0.15956569, 0.08670425, 0.05159195, 0.04564301, 0.161484  ,
        0.09777427, 0.16400941, 0.03993305, 0.16233166, 0.15849606,
        0.06434838, 0.08784175, 0.04214176, 0.16129557, 0.10947029,
        0.16043663, 0.03301565, 0.10177501, 0.16175111, 0.10512431,
        0.0431598 , 0.09208298, 0.0923512 , 0.16087993, 0.08988873,
        0.16194598, 0.15722267, 0.03881772, 0.15991267, 0.08460999,
        0.16373197, 0.04289762, 0.08540471, 0.08215038, 0.07069016,
        0.09411073, 0.10480873, 0.10946361, 0.16592304, 0.05054426,
        0.08244713, 0.04731337, 0.16070875, 0.15845442, 0.0993549 ,
        0.07370567, 0.05057931, 0.04591155, 0.15982167, 0.04799501,
        0.02837745, 0.16248775, 0.15720638, 0.15879067, 0.08094327,
        0.16072226, 0.03561568, 0.16025194, 0.09222968, 0.08663678,
        0.0809261 , 0.16387439, 0.15746808, 0.04599182, 0.16180174,
        0.08571037, 0.07629506, 0.16027069, 0.02929934, 0.04230634,
        0.03476755, 0.06879314, 0.075454  , 0.07781498, 0.03032009,
        0.08300734, 0.08738105, 0.09607434, 0.06780799, 0.0932068 ,
        0.07049743, 0.16685406, 0.1572485 , 0.09049996, 0.04913672,
        0.16436434, 0.07093898, 0.16357732, 0.15857577, 0.02998702,
        0.16222127, 0.15877859, 0.04653668, 0.08763909, 0.10107597,
        0.10169204, 0.0339721 , 0.10015663, 0.04281108, 0.07828856,
        0.0539279 , 0.08881172, 0.16352733, 0.04617365, 0.06986491,
        0.05810459, 0.16255959, 0.15785495, 0.15909561, 0.15884662,
        0.02808722, 0.16196489, 0.04352164, 0.16040699, 0.15682022,
        0.15723006, 0.15645345, 0.03721595, 0.02905973, 0.07962203,
        0.08202394, 0.16780774, 0.10550682, 0.10215012, 0.10173853,
        0.03122362, 0.16161267, 0.03868055, 0.08127626, 0.0985655 ,
        0.05747676, 0.1651744 , 0.04680403, 0.04940311, 0.16158398,
        0.0350879 , 0.09144322, 0.04387879, 0.07343022, 0.11099966,
        0.16437435, 0.05200148, 0.03364499, 0.07398868, 0.0345517 ,
        0.03137175, 0.06525763, 0.0787247 , 0.04402137, 0.08349204,
        0.03217769, 0.16286   , 0.15982588, 0.10864202, 0.08716512,
        0.16332459, 0.08708994, 0.04588413, 0.07602906, 0.16599838,
        0.15694801, 0.15773598, 0.15643803, 0.15715996, 0.15674305,
        0.08549142, 0.08267204, 0.0836846 , 0.10728947, 0.05842018,
        0.11305968, 0.09622701, 0.04364141, 0.06589238, 0.04725289,
        0.05248761, 0.07489793, 0.07797805, 0.08651121, 0.07102553,
        0.16860096, 0.1584332 , 0.04244844, 0.03073438, 0.02634199,
        0.04510172, 0.04973785, 0.03328904, 0.04465024, 0.03434889,
        0.053708  , 0.16599917, 0.15662758, 0.0601    , 0.16615407,
        0.15853969, 0.05609282, 0.07197587, 0.08120163, 0.10609444,
```

```
        0.10673436, 0.16541394, 0.05018409, 0.04155596, 0.16728203,
        0.15968553, 0.09017897, 0.07150563, 0.07796367, 0.0426844 ,
        0.07441966, 0.07822037, 0.16743302, 0.16039371, 0.09663963,
        0.05863865, 0.16694252, 0.15953104, 0.09373013, 0.16678675,
        0.03835972, 0.16223232, 0.08333731, 0.04617731, 0.07907216,
        0.17463334, 0.10899703, 0.16382043, 0.16051563, 0.08369851,
        0.03255987, 0.16586002, 0.10756063, 0.16713365, 0.09712799,
        0.16689507, 0.09904575, 0.08941182, 0.10806767, 0.16353559,
        0.1595703 , 0.03538561, 0.16535266, 0.04760734, 0.0472343 ,
        0.04760965, 0.16728131, 0.04449224, 0.11961206, 0.16532278]),
 'std_fit_time': array([2.11457050e-03, 1.65312408e-02, 2.84375997e-02, 3.20034553e-02,
        3.08402733e-03, 5.12756850e-03, 6.12738800e-03, 1.03250517e-02,
        6.08512998e-03, 9.97849352e-04, 8.66250300e-03, 1.10565803e-02,
        2.43582095e-02, 3.33600893e-03, 1.27077392e-02, 2.95930462e-03,
        1.04896165e-02, 1.16638857e-02, 3.42860602e-03, 1.83782997e-02,
        1.03675511e-02, 3.17292926e-02, 8.63222612e-03, 3.18551703e-03,
        1.73830080e-02, 4.13283837e-03, 3.85248932e-04, 1.94474530e-02,
        3.27475484e-03, 4.30617392e-03, 9.60540346e-03, 7.06356176e-03,
        1.25655424e-02, 3.84139731e-03, 3.57323428e-03, 1.59482145e-02,
        1.93917707e-02, 3.06422750e-02, 8.40788446e-03, 5.46383117e-03,
        5.73864237e-03, 8.18652352e-03, 3.94559429e-03, 2.86325593e-04,
        1.41156771e-02, 1.01689851e-02, 2.08312540e-02, 1.33353553e-02,
        3.46772974e-03, 1.82695632e-02, 2.79750688e-03, 1.47932050e-03,
        3.45888383e-04, 1.46785829e-03, 4.03534269e-03, 3.88406005e-03,
        1.19196764e-02, 4.11445568e-03, 5.90945505e-03, 8.09676546e-03,
        4.17451707e-03, 9.49324036e-03, 8.65078173e-04, 1.34172331e-02,
        3.72610415e-03, 4.35425724e-03, 3.24082890e-02, 3.00587884e-03,
        1.14665945e-02, 1.53872127e-02, 1.01511425e-02, 6.30203959e-03,
        4.23127622e-03, 5.48394079e-03, 5.46581933e-03, 1.11189569e-02,
        1.38173217e-02, 9.94262541e-03, 7.38262618e-03, 2.29021486e-02,
        2.36126352e-02, 1.13492272e-02, 8.17896062e-04, 2.65255245e-02,
        3.10009323e-02, 9.70221940e-03, 2.54712108e-02, 6.73088274e-03,
        8.59645114e-04, 7.59393210e-03, 6.01402728e-03, 8.82039285e-04,
        2.84241413e-02, 1.34325225e-02, 1.40466124e-02, 4.94366557e-03,
        1.18203431e-02, 1.55759654e-02, 7.25026568e-03, 5.83343830e-03,
        2.08706605e-02, 9.24877189e-03, 6.75400760e-03, 1.98692281e-02,
        2.81544372e-02, 9.88633288e-03, 2.20734057e-03, 5.11365792e-04,
        3.72433750e-04, 8.48350953e-04, 5.46952357e-03, 2.33560730e-03,
        1.12439143e-02, 2.77701554e-03, 4.32182209e-04, 6.34917743e-04,
        4.19376421e-04, 2.24295357e-03, 1.14709505e-02, 9.94475448e-03,
        1.46756442e-02, 1.07489293e-02, 1.95442321e-02, 7.70290633e-03,
        6.58605171e-03, 8.97865487e-03, 3.67849319e-03, 2.35450452e-03,
        2.11331394e-02, 1.81515079e-02, 3.69647007e-02, 4.46778414e-03,
        1.07359664e-02, 4.90675560e-03, 3.52560175e-03, 6.16379191e-03,
        9.41083718e-03, 2.20919687e-03, 4.90169946e-03, 3.71866142e-03,
        5.19583865e-03, 1.84697354e-02, 5.35574733e-03, 9.13684066e-03,
        7.30151906e-03, 4.70719559e-03, 2.30510013e-02, 5.72698063e-03,
        2.02976201e-02, 5.10548646e-03, 8.80015979e-03, 3.96344885e-03,
        9.57369526e-04, 6.04881609e-03, 2.32945814e-02, 4.98227796e-03,
        1.96885968e-02, 1.76036320e-02, 8.22221400e-03, 6.07503783e-03,
        4.14467429e-04, 1.14410357e-03, 2.55527799e-04, 7.17401711e-04,
        7.66982551e-04, 1.01090403e-02, 1.17314807e-02, 6.12455699e-03,
        2.44046944e-02, 1.74684305e-02, 1.52099576e-02, 9.89410510e-03,
        3.31422038e-02, 3.90448068e-02, 1.47390534e-02, 1.88397155e-02,
        1.06614448e-02, 3.99230960e-03, 7.08130921e-03, 2.99728882e-03,
        4.97825961e-03, 1.57480568e-03, 9.56956383e-03, 7.60305070e-03,
        7.81736009e-03, 1.53879414e-02, 2.40258364e-02, 3.43276891e-03,
        5.64806513e-03, 1.44651814e-02, 2.09718080e-02, 1.86487315e-03,
        8.91941151e-05, 2.28768798e-02, 1.09586585e-02, 1.74209074e-03,
        3.01223680e-02, 9.08934146e-03, 7.12856086e-03, 9.21045435e-03,
        5.23643382e-03, 4.80507744e-03, 3.00736883e-02, 4.63081261e-03,
        6.03316278e-03, 2.83427298e-03, 2.00829174e-02, 1.13148198e-02,
        4.02999139e-03, 5.45059756e-03, 2.30067126e-03, 1.06264524e-02,
        5.40292893e-03, 2.90689405e-03, 6.52718154e-03, 3.04120914e-02,
        7.93194876e-03, 1.00059407e-03, 4.03543901e-03, 7.03970894e-03,
        2.15658094e-02, 5.26499343e-03, 7.58986881e-03, 1.45468669e-02,
        5.25142420e-03, 5.10550387e-03, 1.45709798e-02, 1.98346399e-03,
        4.89606676e-04, 8.93667131e-03, 2.55383063e-03, 4.20919387e-03,
        1.34881402e-02, 7.80405782e-03, 3.07381369e-03, 5.03112975e-03,
        4.56187553e-03, 1.38241600e-02, 2.10477478e-03, 2.66418305e-03,
```

```
       2.12256126e-03, 5.52015188e-03, 3.96064984e-03, 1.79567921e-02,
       6.36661377e-03, 5.98138318e-03, 2.63866863e-03, 9.21107446e-03,
       3.37387012e-03, 1.75406065e-03]),
 'mean_score_time': array([0.00166233, 0.00141001, 0.00137067, 0.00129795, 0.00176032,
       0.0018727 , 0.00179513, 0.00140055, 0.0016923 , 0.0017643 ,
       0.00153732, 0.00143758, 0.00137281, 0.00173537, 0.00159208,
       0.00176597, 0.0014507 , 0.00164294, 0.00180332, 0.00147772,
       0.00157094, 0.00155433, 0.00148606, 0.00171638, 0.00148002,
       0.00187016, 0.00167616, 0.00155997, 0.00171638, 0.00138243,
       0.00160837, 0.00128762, 0.00139793, 0.00221793, 0.00137154,
       0.00142392, 0.00151404, 0.00151396, 0.00185362, 0.00193961,
       0.00151571, 0.00156999, 0.00180761, 0.00172536, 0.00149417,
       0.00196759, 0.00127832, 0.00146476, 0.0016547 , 0.00137981,
       0.00149488, 0.00171065, 0.00168729, 0.00178798, 0.00144998,
       0.00172869, 0.00247232, 0.00174745, 0.00182033, 0.00143488,
       0.00137726, 0.00164294, 0.0016818 , 0.00233833, 0.00169698,
       0.00151531, 0.00130296, 0.00185633, 0.00128309, 0.00172003,
       0.00137202, 0.0012846 , 0.00146421, 0.00132362, 0.00235295,
       0.00186388, 0.00148193, 0.00148201, 0.00128794, 0.00146159,
       0.00161099, 0.00169754, 0.00156228, 0.00142272, 0.00137329,
       0.00166297, 0.00132442, 0.00171932, 0.00176318, 0.0013744 ,
       0.00163301, 0.00179235, 0.00139475, 0.00153772, 0.00148416,
       0.00150172, 0.00136328, 0.00188128, 0.00146866, 0.00126998,
       0.00154233, 0.00153891, 0.00156355, 0.00186189, 0.00194701,
       0.00144688, 0.00191601, 0.00160042, 0.00185426, 0.00175373,
       0.00152246, 0.0018177 , 0.00156236, 0.00181063, 0.00155298,
       0.00175865, 0.00152818, 0.00137464, 0.0012811 , 0.00168427,
       0.0013241 , 0.00174403, 0.00154781, 0.00157102, 0.00160265,
       0.00140699, 0.00164636, 0.00145404, 0.001525  , 0.00140897,
       0.00163356, 0.00188057, 0.00163229, 0.00156887, 0.00186324,
       0.00155632, 0.0015192 , 0.00142654, 0.00128921, 0.00151881,
       0.00173434, 0.00131639, 0.00143003, 0.00132187, 0.00126521,
       0.00125066, 0.00133936, 0.00133562, 0.00137019, 0.00136065,
       0.00132259, 0.00169603, 0.00177169, 0.0019904 , 0.00218312,
       0.00180308, 0.00147502, 0.00129882, 0.00137329, 0.00153724,
       0.00160201, 0.00166035, 0.0015409 , 0.00165637, 0.00155369,
       0.00140103, 0.00218463, 0.00297236, 0.00153867, 0.00137814,
       0.00159629, 0.00154034, 0.00129159, 0.0013682 , 0.00194184,
       0.00144267, 0.0013748 , 0.00134722, 0.00213337, 0.00134174,
       0.00160257, 0.00149242, 0.00143027, 0.00124963, 0.00136685,
       0.00125806, 0.00132902, 0.00126561, 0.00126974, 0.00124041,
       0.00196648, 0.00153454, 0.00140882, 0.00146898, 0.0015076 ,
       0.00158993, 0.0013636 , 0.00128531, 0.0015173 , 0.00171995,
       0.00170294, 0.00172504, 0.00142996, 0.00139268, 0.00172091,
       0.00148424, 0.00139594, 0.00130924, 0.00140134, 0.00127506,
       0.00136423, 0.0021716 , 0.00173338, 0.00160066, 0.00140961,
       0.00172933, 0.00161664, 0.00177304, 0.00148447, 0.00179267,
       0.00131138, 0.001635  , 0.00146476, 0.00129302, 0.00149417,
       0.00160845, 0.00163309, 0.00193834, 0.00190632, 0.00140651,
       0.00138783, 0.00179195, 0.00158525, 0.00185998, 0.00160901,
       0.00181007, 0.00176191, 0.00148813, 0.00159939, 0.0018301 ,
       0.00160376, 0.00157642, 0.00184162, 0.00162737, 0.0017097 ,
       0.0015409 , 0.00193119, 0.00159971, 0.00167998, 0.00179521]),
 'std_score_time': array([1.57548846e-04, 9.89922439e-05, 9.90452494e-05, 2.53539598e-05,
       1.16986587e-04, 5.40891746e-04, 8.75290279e-05, 1.25420310e-04,
       2.60685832e-04, 3.13325194e-04, 2.15800645e-05, 9.72195102e-05,
       1.39544068e-04, 7.71465772e-05, 8.48514123e-05, 1.03580366e-04,
       8.35124013e-05, 3.20994176e-04, 2.14518639e-04, 1.01556986e-04,
       1.79253145e-05, 6.77515335e-05, 1.04524184e-05, 1.78763806e-04,
       1.11140969e-04, 8.28245234e-05, 1.19726384e-04, 3.63544686e-05,
       1.93694908e-04, 1.00687252e-04, 2.29675272e-04, 3.33401671e-05,
       8.63922990e-05, 1.28432520e-03, 7.42454324e-05, 6.45453693e-05,
       2.49876118e-05, 1.40133347e-04, 2.05393912e-05, 8.40435858e-04,
       1.71863619e-05, 2.68408921e-05, 2.17720276e-04, 3.63963139e-05,
       5.12875254e-05, 7.86598318e-04, 1.80321099e-05, 5.23860593e-05,
       2.57738485e-04, 1.29982424e-04, 2.08403843e-05, 7.49946768e-05,
       8.83322871e-05, 7.76862549e-05, 8.64032644e-05, 1.09800377e-04,
       1.84143366e-04, 6.51647404e-05, 4.22583329e-04, 1.11573329e-04,
       8.68520921e-05, 1.02796477e-04, 6.79725752e-05, 1.14459840e-03,
       7.04160931e-05, 5.34909615e-05, 3.24220048e-05, 1.53341889e-04,
```

```
        1.78589503e-05, 6.44119179e-04, 7.76064581e-05, 1.12234137e-05,
        1.45461431e-04, 1.54197829e-05, 9.94995430e-04, 6.91447431e-04,
        1.28784635e-04, 6.50747341e-05, 1.58878153e-05, 9.11653816e-05,
        1.55458177e-04, 7.47806249e-05, 1.89898798e-04, 8.71635051e-05,
        8.43033621e-05, 1.11466912e-04, 8.28766667e-05, 1.37057070e-04,
        4.04020822e-05, 9.22375581e-05, 1.88500242e-04, 8.79917100e-05,
        6.96625742e-05, 1.95323401e-04, 9.92361786e-05, 3.52560452e-05,
        6.25496341e-05, 4.57739843e-04, 8.74862999e-05, 2.88957897e-06,
        3.68930443e-05, 1.90990231e-04, 1.66852242e-04, 5.35778637e-04,
        4.64560557e-04, 1.07862444e-04, 7.19190301e-05, 1.61076450e-04,
        5.21220798e-05, 7.26362808e-05, 8.27937100e-05, 6.50537665e-05,
        5.40546327e-05, 2.09318780e-04, 1.87654692e-04, 6.55127326e-05,
        1.91475734e-04, 9.16475422e-05, 2.91887272e-05, 4.76166625e-04,
        2.40052769e-05, 1.19510271e-04, 4.47909771e-05, 2.73384327e-05,
        6.03461240e-05, 1.05142563e-04, 9.61459359e-05, 1.34744587e-04,
        1.49455469e-05, 7.57529362e-05, 9.36426382e-05, 1.33461081e-04,
        5.48429531e-05, 7.34510681e-05, 2.30147008e-04, 5.35338055e-05,
        2.08021633e-05, 8.57384357e-05, 2.24133191e-05, 2.69615703e-05,
        1.47964337e-04, 4.38371891e-05, 1.32222539e-04, 3.06886688e-05,
        3.89984263e-06, 4.18573827e-06, 1.02691790e-04, 9.50840873e-05,
        6.89528625e-05, 9.35871127e-05, 8.10979160e-05, 9.04670668e-05,
        4.85277967e-05, 7.01173228e-04, 1.12032949e-03, 5.51791231e-05,
        4.12917913e-05, 2.82800573e-05, 1.45166962e-04, 1.71707436e-04,
        1.47271980e-04, 1.95561765e-04, 1.92115476e-04, 1.13347969e-04,
        2.10592612e-04, 7.30794322e-05, 1.11655503e-03, 1.17718696e-03,
        3.53933610e-05, 6.57511970e-05, 1.16729751e-04, 2.93492744e-05,
        3.69363314e-05, 1.04413970e-04, 5.12080570e-04, 8.60628327e-05,
        1.39605202e-04, 1.06802712e-04, 1.16521816e-03, 8.60089494e-05,
        1.86231654e-04, 2.13638775e-04, 9.54383348e-05, 9.67938976e-06,
        1.32207491e-04, 9.84244978e-06, 1.15426826e-04, 1.42644375e-05,
        9.26327689e-06, 9.28983001e-06, 8.42148039e-04, 1.85032104e-04,
        7.30980114e-05, 1.04448806e-04, 8.56114903e-05, 2.28639509e-04,
        7.07484341e-05, 5.84976394e-06, 2.83451783e-04, 2.83929668e-04,
        2.66684962e-04, 9.16785489e-05, 1.14966275e-04, 9.16835090e-05,
        5.73940426e-05, 2.14768812e-04, 7.67181939e-05, 1.98430856e-05,
        9.05742388e-05, 9.28506966e-06, 9.45150510e-05, 1.15518337e-03,
        1.39667757e-04, 1.87754425e-04, 8.20530240e-05, 3.10277291e-04,
        2.40479253e-04, 5.09153778e-05, 3.70919512e-05, 1.31728512e-05,
        5.91256712e-05, 1.82445455e-04, 5.04548145e-05, 2.53070837e-05,
        6.79508287e-05, 1.71679736e-04, 1.35780352e-04, 1.13841813e-04,
        1.80235861e-04, 8.70199145e-05, 9.67640731e-05, 9.39655583e-05,
        2.18857434e-04, 1.02897130e-04, 1.46591864e-04, 8.54410493e-05,
        2.72317021e-04, 1.32966241e-04, 7.64235581e-05, 6.79257280e-05,
        1.82187193e-04, 3.72197799e-05, 1.23199185e-04, 1.03836941e-04,
        1.93169768e-04, 9.20513931e-06, 2.33164675e-04, 3.16192138e-05,
        1.59521609e-04, 1.20090775e-04]),
'param_warm_start': masked_array(data=[False, True, False, False, False, True, True, False,
                   False, True, True, False, False, False, True, False,
                   False, True, True, False, True, False, True, True,
                   False, False, False, True, False, False, False, True,
                   False, True, True, False, False, False, False, False,
                   False, False, True, True, True, True, False, False,
                   False, False, False, False, False, False, False, False,
                   True, True, False, True, False, False, True, True,
                   False, True, True, False, False, True, False, True,
                   True, True, True, True, True, True, False, False,
                   False, True, True, True, True, True, True, False, True,
                   False, True, False, True, False, True, False, False,
                   True, True, False, True, False, True, True, False,
                   False, True, True, True, True, True, False, False,
                   True, True, False, False, False, True, True, True,
                   False, True, False, False, False, True, False, True,
                   False, False, False, False, True, False, False, True,
                   True, False, True, True, False, True, True, True,
                   False, False, True, False, False, False, True, True,
                   False, False, True, False, True, True, False, True,
                   False, True, False, True, True, True, True, True,
                   False, False, False, True, True, True, False, False,
                   True, True, False, False, True, True, True, True, True,
                   True, True, False, False, True, True, True, True, True,
```

```
                    True, True, False, False, True, True, True, False,
                    True, True, False, False, True, False, True, False,
                    False, True, False, False, True, False, False, True,
                    True, False, True, True, False, False, False, False,
                    True, False, True, False, True, True, True, False,
                    True, True, False, False, False, False, True, False,
                    True, False, False, False, True, False, False],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'param_solver': masked_array(data=['sag', 'newton-cg', 'lbfgs', 'lbfgs', 'sag',
                    'newton-cg', 'sag', 'lbfgs', 'sag', 'sag', 'lbfgs',
                    'newton-cg', 'lbfgs', 'sag', 'newton-cg', 'sag',
                    'lbfgs', 'newton-cg', 'sag', 'newton-cg', 'lbfgs',
                    'lbfgs', 'newton-cg', 'sag', 'newton-cg', 'sag', 'sag',
                    'lbfgs', 'sag', 'newton-cg', 'sag', 'lbfgs',
                    'newton-cg', 'newton-cg', 'newton-cg', 'newton-cg',
                    'newton-cg', 'newton-cg', 'sag', 'lbfgs', 'newton-cg',
                    'lbfgs', 'sag', 'sag', 'newton-cg', 'newton-cg',
                    'lbfgs', 'lbfgs', 'sag', 'lbfgs', 'lbfgs', 'sag',
                    'sag', 'sag', 'newton-cg', 'sag', 'lbfgs', 'sag',
                    'newton-cg', 'newton-cg', 'newton-cg', 'sag', 'sag',
                    'lbfgs', 'sag', 'newton-cg', 'lbfgs', 'sag', 'lbfgs',
                    'lbfgs', 'lbfgs', 'newton-cg', 'newton-cg',
                    'newton-cg', 'lbfgs', 'newton-cg', 'newton-cg',
                    'newton-cg', 'newton-cg', 'newton-cg', 'lbfgs', 'sag',
                    'sag', 'newton-cg', 'lbfgs', 'sag', 'lbfgs', 'sag',
                    'sag', 'lbfgs', 'sag', 'sag', 'lbfgs', 'newton-cg',
                    'newton-cg', 'newton-cg', 'lbfgs', 'newton-cg',
                    'lbfgs', 'newton-cg', 'lbfgs', 'newton-cg', 'sag',
                    'lbfgs', 'lbfgs', 'lbfgs', 'sag', 'sag', 'sag', 'sag',
                    'lbfgs', 'sag', 'lbfgs', 'sag', 'sag', 'sag', 'sag',
                    'lbfgs', 'lbfgs', 'newton-cg', 'newton-cg', 'sag',
                    'newton-cg', 'newton-cg', 'newton-cg', 'lbfgs', 'sag',
                    'lbfgs', 'lbfgs', 'newton-cg', 'lbfgs', 'sag', 'lbfgs',
                    'lbfgs', 'sag', 'lbfgs', 'newton-cg', 'lbfgs',
                    'newton-cg', 'newton-cg', 'sag', 'lbfgs', 'lbfgs',
                    'newton-cg', 'lbfgs', 'lbfgs', 'lbfgs', 'newton-cg',
                    'lbfgs', 'newton-cg', 'lbfgs', 'sag', 'sag',
```

```
                 'newton-cg', 'newton-cg', 'sag', 'newton-cg', 'lbfgs',
                 'newton-cg', 'sag', 'sag', 'sag', 'sag', 'sag', 'sag',
                 'newton-cg', 'newton-cg', 'newton-cg', 'newton-cg',
                 'lbfgs', 'newton-cg', 'newton-cg', 'lbfgs', 'lbfgs',
                 'lbfgs', 'lbfgs', 'newton-cg', 'newton-cg',
                 'newton-cg', 'newton-cg', 'sag', 'sag', 'lbfgs',
                 'lbfgs', 'lbfgs', 'lbfgs', 'lbfgs', 'lbfgs', 'lbfgs',
                 'lbfgs', 'lbfgs', 'sag', 'sag', 'lbfgs', 'sag', 'sag',
                 'lbfgs', 'newton-cg', 'newton-cg', 'newton-cg',
                 'newton-cg', 'sag', 'lbfgs', 'lbfgs', 'sag', 'sag',
                 'newton-cg', 'newton-cg', 'newton-cg', 'lbfgs',
                 'newton-cg', 'newton-cg', 'sag', 'sag', 'newton-cg',
                 'lbfgs', 'sag', 'sag', 'newton-cg', 'sag', 'lbfgs',
                 'sag', 'newton-cg', 'lbfgs', 'newton-cg', 'sag',
                 'newton-cg', 'sag', 'sag', 'newton-cg', 'lbfgs', 'sag',
                 'newton-cg', 'sag', 'newton-cg', 'sag', 'newton-cg',
                 'newton-cg', 'newton-cg', 'sag', 'sag', 'lbfgs', 'sag',
                 'lbfgs', 'lbfgs', 'lbfgs', 'sag', 'lbfgs', 'newton-cg',
                 'sag'],
           mask=[False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False, False, False, False, False, False, False,
                 False, False],
     fill_value='?',
          dtype=object),
 'param_C': masked_array(data=[1.4000000000000001, 3.0000000000000004,
                 2.4000000000000004, 3.0000000000000004,
                 1.3000000000000003, 3.9000000000000004,
                 1.4000000000000001, 0.2, 3.3000000000000003, 4.3,
                 1.5000000000000002, 1.8000000000000003, 0.9,
                 3.9000000000000004, 4.0, 0.30000000000000004,
                 1.4000000000000001, 3.6, 0.8, 2.5000000000000004,
                 2.8000000000000003, 4.8, 2.9000000000000004, 2.1,
                 3.8000000000000003, 1.2000000000000002,
                 1.5000000000000002, 2.3000000000000003, 0.5,
                 4.3999999999999995, 0.7000000000000001,
                 4.3999999999999995, 1.0, 1.0, 3.5000000000000004,
                 2.4000000000000004, 0.7000000000000001, 4.0, 1.0,
                 1.5000000000000002, 0.8, 1.8000000000000003, 0.1,
                 1.3000000000000003, 1.1, 1.2000000000000002, 4.7,
                 0.30000000000000004, 3.8000000000000003,
                 4.3999999999999995, 0.7000000000000001, 0.9, 0.8, 2.7,
```

```
          2.7, 3.7, 1.4000000000000001, 2.2, 0.4,
          3.3000000000000003, 3.9000000000000004, 2.0, 0.5, 4.1,
          1.7000000000000002, 1.6, 3.4000000000000004, 0.4,
          2.3000000000000003, 0.30000000000000004, 2.6, 0.6,
          1.8000000000000003, 2.1, 1.7000000000000002, 0.5, 0.2,
          2.2, 3.5000000000000004, 4.3, 0.5, 0.2, 0.4,
          1.3000000000000003, 0.6, 3.2, 2.9000000000000004, 4.1,
          1.0, 0.4, 3.7, 2.4000000000000004, 3.0000000000000004,
          2.9000000000000004, 2.7, 4.6, 3.5000000000000004,
          1.9000000000000001, 2.6, 3.0000000000000004, 1.0, 2.0,
          0.30000000000000004, 0.1, 4.0, 1.1, 2.3000000000000003,
          3.9000000000000004, 3.1, 0.9, 0.7000000000000001, 4.8,
          1.3000000000000003, 2.9000000000000004,
          0.7000000000000001, 3.2, 4.3, 1.9000000000000001, 4.6,
          3.2, 0.8, 2.6, 2.3000000000000003, 0.5, 2.1, 4.6, 4.8,
          1.7000000000000002, 4.8, 3.6, 1.2000000000000002, 2.1,
          2.1, 1.9000000000000001, 0.2, 4.5, 3.4000000000000004,
          1.1, 0.6, 4.7, 1.2000000000000002, 3.3000000000000003,
          2.1, 1.5000000000000002, 1.6, 3.6, 2.9000000000000004,
          2.8000000000000003, 2.7, 4.1, 4.1, 2.4000000000000004,
          4.0, 4.2, 1.3000000000000003, 1.5000000000000002,
          1.9000000000000001, 4.7, 4.2, 3.5000000000000004,
          1.9000000000000001, 3.6, 4.5, 1.8000000000000003, 4.6,
          3.7, 2.5000000000000004, 4.5, 1.4000000000000001, 0.8,
          2.6, 2.3000000000000003, 1.2000000000000002,
          2.4000000000000004, 0.9, 4.2, 0.1, 4.3999999999999995,
          2.6, 1.6, 0.1, 2.8000000000000003, 0.2,
          1.3000000000000003, 4.5, 2.5000000000000004, 4.2, 2.2,
          4.3, 1.0, 3.8000000000000003, 2.6, 2.5000000000000004,
          0.5, 1.7000000000000002, 4.9, 3.9000000000000004, 4.9,
          3.2, 0.4, 4.1, 3.0000000000000004, 0.6, 3.2, 4.2, 1.6,
          1.4000000000000001, 0.1, 0.30000000000000004, 4.3,
          1.2000000000000002, 0.9, 2.7, 2.9000000000000004, 4.7,
          4.0, 2.3000000000000003, 4.0, 2.0, 1.8000000000000003,
          0.1, 3.6, 4.9, 2.5000000000000004, 3.4000000000000004,
          2.8000000000000003, 3.3000000000000003,
          3.8000000000000003, 3.1, 0.9, 3.2, 1.1,
          3.8000000000000003, 3.5000000000000004, 3.1, 4.1, 4.8,
          0.2, 3.7, 3.4000000000000004, 4.6, 4.9, 4.2,
          3.5000000000000004, 2.2, 3.1, 1.1, 1.8000000000000003,
          4.8, 4.9],
    mask=[False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
          False, False, False, False, False, False, False, False,
```

```
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False],
              fill_value='?',
                   dtype=object),
 'params': [{'warm_start': False, 'solver': 'sag', 'C': 1.4000000000000001},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 3.0000000000000004},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 2.4000000000000004},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 3.0000000000000004},
  {'warm_start': False, 'solver': 'sag', 'C': 1.3000000000000003},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 3.9000000000000004},
  {'warm_start': True, 'solver': 'sag', 'C': 1.4000000000000001},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 0.2},
  {'warm_start': False, 'solver': 'sag', 'C': 3.3000000000000003},
  {'warm_start': True, 'solver': 'sag', 'C': 4.3},
  {'warm_start': True, 'solver': 'lbfgs', 'C': 1.5000000000000002},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 1.8000000000000003},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 0.9},
  {'warm_start': False, 'solver': 'sag', 'C': 3.9000000000000004},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 4.0},
  {'warm_start': False, 'solver': 'sag', 'C': 0.30000000000000004},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 1.4000000000000001},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 3.6},
  {'warm_start': True, 'solver': 'sag', 'C': 0.8},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 2.5000000000000004},
  {'warm_start': True, 'solver': 'lbfgs', 'C': 2.8000000000000003},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 4.8},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 2.9000000000000004},
  {'warm_start': True, 'solver': 'sag', 'C': 2.1},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 3.8000000000000003},
  {'warm_start': False, 'solver': 'sag', 'C': 1.2000000000000002},
  {'warm_start': False, 'solver': 'sag', 'C': 1.5000000000000002},
  {'warm_start': True, 'solver': 'lbfgs', 'C': 2.3000000000000003},
  {'warm_start': False, 'solver': 'sag', 'C': 0.5},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 4.3999999999999995},
  {'warm_start': False, 'solver': 'sag', 'C': 0.7000000000000001},
  {'warm_start': True, 'solver': 'lbfgs', 'C': 4.3999999999999995},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 1.0},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 1.0},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 3.5000000000000004},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 2.4000000000000004},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 0.7000000000000001},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 4.0},
  {'warm_start': False, 'solver': 'sag', 'C': 1.0},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 1.5000000000000002},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 0.8},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 1.8000000000000003},
  {'warm_start': True, 'solver': 'sag', 'C': 0.1},
  {'warm_start': True, 'solver': 'sag', 'C': 1.3000000000000003},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 1.1},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 1.2000000000000002},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 4.7},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 0.30000000000000004},
  {'warm_start': False, 'solver': 'sag', 'C': 3.8000000000000003},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 4.3999999999999995},
  {'warm_start': False, 'solver': 'lbfgs', 'C': 0.7000000000000001},
  {'warm_start': False, 'solver': 'sag', 'C': 0.9},
  {'warm_start': False, 'solver': 'sag', 'C': 0.8},
  {'warm_start': False, 'solver': 'sag', 'C': 2.7},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 2.7},
  {'warm_start': False, 'solver': 'sag', 'C': 3.7},
  {'warm_start': True, 'solver': 'lbfgs', 'C': 1.4000000000000001},
  {'warm_start': True, 'solver': 'sag', 'C': 2.2},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 0.4},
  {'warm_start': True, 'solver': 'newton-cg', 'C': 3.3000000000000003},
  {'warm_start': False, 'solver': 'newton-cg', 'C': 3.9000000000000004},
  {'warm_start': False, 'solver': 'sag', 'C': 2.0},
  {'warm_start': True, 'solver': 'sag', 'C': 0.5},
  {'warm_start': True, 'solver': 'lbfgs', 'C': 4.1},
  {'warm_start': False, 'solver': 'sag', 'C': 1.7000000000000002},
```

```
{'warm_start': True, 'solver': 'newton-cg', 'C': 1.6},
{'warm_start': True, 'solver': 'lbfgs', 'C': 3.4000000000000004},
{'warm_start': False, 'solver': 'sag', 'C': 0.4},
{'warm_start': False, 'solver': 'lbfgs', 'C': 2.3000000000000003},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.30000000000000004},
{'warm_start': False, 'solver': 'lbfgs', 'C': 2.6},
{'warm_start': True, 'solver': 'newton-cg', 'C': 0.6},
{'warm_start': True, 'solver': 'newton-cg', 'C': 1.8000000000000003},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.1},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.7000000000000002},
{'warm_start': True, 'solver': 'newton-cg', 'C': 0.5},
{'warm_start': True, 'solver': 'newton-cg', 'C': 0.2},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.2},
{'warm_start': False, 'solver': 'newton-cg', 'C': 3.5000000000000004},
{'warm_start': False, 'solver': 'newton-cg', 'C': 4.3},
{'warm_start': False, 'solver': 'lbfgs', 'C': 0.5},
{'warm_start': True, 'solver': 'sag', 'C': 0.2},
{'warm_start': True, 'solver': 'sag', 'C': 0.4},
{'warm_start': True, 'solver': 'newton-cg', 'C': 1.3000000000000003},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.6},
{'warm_start': True, 'solver': 'sag', 'C': 3.2},
{'warm_start': True, 'solver': 'lbfgs', 'C': 2.9000000000000004},
{'warm_start': False, 'solver': 'sag', 'C': 4.1},
{'warm_start': True, 'solver': 'sag', 'C': 1.0},
{'warm_start': False, 'solver': 'lbfgs', 'C': 0.4},
{'warm_start': True, 'solver': 'sag', 'C': 3.7},
{'warm_start': False, 'solver': 'sag', 'C': 2.4000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 3.0000000000000004},
{'warm_start': False, 'solver': 'newton-cg', 'C': 2.9000000000000004},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.7},
{'warm_start': False, 'solver': 'newton-cg', 'C': 4.6},
{'warm_start': False, 'solver': 'lbfgs', 'C': 3.5000000000000004},
{'warm_start': True, 'solver': 'newton-cg', 'C': 1.9000000000000001},
{'warm_start': True, 'solver': 'lbfgs', 'C': 2.6},
{'warm_start': False, 'solver': 'newton-cg', 'C': 3.0000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.0},
{'warm_start': False, 'solver': 'newton-cg', 'C': 2.0},
{'warm_start': True, 'solver': 'sag', 'C': 0.30000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.1},
{'warm_start': False, 'solver': 'lbfgs', 'C': 4.0},
{'warm_start': False, 'solver': 'lbfgs', 'C': 1.1},
{'warm_start': True, 'solver': 'sag', 'C': 2.3000000000000003},
{'warm_start': True, 'solver': 'sag', 'C': 3.9000000000000004},
{'warm_start': True, 'solver': 'sag', 'C': 3.1},
{'warm_start': True, 'solver': 'sag', 'C': 0.9},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.7000000000000001},
{'warm_start': False, 'solver': 'sag', 'C': 4.8},
{'warm_start': False, 'solver': 'lbfgs', 'C': 1.3000000000000003},
{'warm_start': True, 'solver': 'sag', 'C': 2.9000000000000004},
{'warm_start': True, 'solver': 'sag', 'C': 0.7000000000000001},
{'warm_start': False, 'solver': 'sag', 'C': 3.2},
{'warm_start': False, 'solver': 'sag', 'C': 4.3},
{'warm_start': False, 'solver': 'lbfgs', 'C': 1.9000000000000001},
{'warm_start': True, 'solver': 'lbfgs', 'C': 4.6},
{'warm_start': True, 'solver': 'newton-cg', 'C': 3.2},
{'warm_start': True, 'solver': 'newton-cg', 'C': 0.8},
{'warm_start': False, 'solver': 'sag', 'C': 2.6},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.3000000000000003},
{'warm_start': False, 'solver': 'newton-cg', 'C': 0.5},
{'warm_start': False, 'solver': 'newton-cg', 'C': 2.1},
{'warm_start': False, 'solver': 'lbfgs', 'C': 4.6},
{'warm_start': True, 'solver': 'sag', 'C': 4.8},
{'warm_start': False, 'solver': 'lbfgs', 'C': 1.7000000000000002},
{'warm_start': True, 'solver': 'lbfgs', 'C': 4.8},
{'warm_start': False, 'solver': 'newton-cg', 'C': 3.6},
{'warm_start': False, 'solver': 'lbfgs', 'C': 1.2000000000000002},
{'warm_start': False, 'solver': 'sag', 'C': 2.1},
{'warm_start': False, 'solver': 'lbfgs', 'C': 2.1},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.9000000000000001},
{'warm_start': False, 'solver': 'sag', 'C': 0.2},
```

```
{'warm_start': False, 'solver': 'lbfgs', 'C': 4.5},
{'warm_start': True, 'solver': 'newton-cg', 'C': 3.4000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.1},
{'warm_start': False, 'solver': 'newton-cg', 'C': 0.6},
{'warm_start': True, 'solver': 'newton-cg', 'C': 4.7},
{'warm_start': True, 'solver': 'sag', 'C': 1.2000000000000002},
{'warm_start': False, 'solver': 'lbfgs', 'C': 3.3000000000000003},
{'warm_start': True, 'solver': 'lbfgs', 'C': 2.1},
{'warm_start': True, 'solver': 'newton-cg', 'C': 1.5000000000000002},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.6},
{'warm_start': False, 'solver': 'lbfgs', 'C': 3.6},
{'warm_start': False, 'solver': 'lbfgs', 'C': 2.9000000000000004},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.8000000000000003},
{'warm_start': False, 'solver': 'lbfgs', 'C': 2.7},
{'warm_start': False, 'solver': 'newton-cg', 'C': 4.1},
{'warm_start': False, 'solver': 'lbfgs', 'C': 4.1},
{'warm_start': True, 'solver': 'sag', 'C': 2.4000000000000004},
{'warm_start': True, 'solver': 'sag', 'C': 4.0},
{'warm_start': False, 'solver': 'newton-cg', 'C': 4.2},
{'warm_start': False, 'solver': 'newton-cg', 'C': 1.3000000000000003},
{'warm_start': True, 'solver': 'sag', 'C': 1.5000000000000002},
{'warm_start': False, 'solver': 'newton-cg', 'C': 1.9000000000000001},
{'warm_start': True, 'solver': 'lbfgs', 'C': 4.7},
{'warm_start': True, 'solver': 'newton-cg', 'C': 4.2},
{'warm_start': False, 'solver': 'sag', 'C': 3.5000000000000004},
{'warm_start': True, 'solver': 'sag', 'C': 1.9000000000000001},
{'warm_start': False, 'solver': 'sag', 'C': 3.6},
{'warm_start': True, 'solver': 'sag', 'C': 4.5},
{'warm_start': False, 'solver': 'sag', 'C': 1.8000000000000003},
{'warm_start': True, 'solver': 'sag', 'C': 4.6},
{'warm_start': True, 'solver': 'newton-cg', 'C': 3.7},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.5000000000000004},
{'warm_start': True, 'solver': 'newton-cg', 'C': 4.5},
{'warm_start': True, 'solver': 'newton-cg', 'C': 1.4000000000000001},
{'warm_start': False, 'solver': 'lbfgs', 'C': 0.8},
{'warm_start': False, 'solver': 'newton-cg', 'C': 2.6},
{'warm_start': False, 'solver': 'newton-cg', 'C': 2.3000000000000003},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.2000000000000002},
{'warm_start': True, 'solver': 'lbfgs', 'C': 2.4000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.9},
{'warm_start': False, 'solver': 'lbfgs', 'C': 4.2},
{'warm_start': False, 'solver': 'newton-cg', 'C': 0.1},
{'warm_start': True, 'solver': 'newton-cg', 'C': 4.3999999999999995},
{'warm_start': True, 'solver': 'newton-cg', 'C': 2.6},
{'warm_start': False, 'solver': 'newton-cg', 'C': 1.6},
{'warm_start': False, 'solver': 'sag', 'C': 0.1},
{'warm_start': True, 'solver': 'sag', 'C': 2.8000000000000003},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.2},
{'warm_start': True, 'solver': 'lbfgs', 'C': 1.3000000000000003},
{'warm_start': True, 'solver': 'lbfgs', 'C': 4.5},
{'warm_start': True, 'solver': 'lbfgs', 'C': 2.5000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 4.2},
{'warm_start': True, 'solver': 'lbfgs', 'C': 2.2},
{'warm_start': False, 'solver': 'lbfgs', 'C': 4.3},
{'warm_start': False, 'solver': 'lbfgs', 'C': 1.0},
{'warm_start': True, 'solver': 'lbfgs', 'C': 3.8000000000000003},
{'warm_start': True, 'solver': 'sag', 'C': 2.6},
{'warm_start': True, 'solver': 'sag', 'C': 2.5000000000000004},
{'warm_start': True, 'solver': 'lbfgs', 'C': 0.5},
{'warm_start': True, 'solver': 'sag', 'C': 1.7000000000000002},
{'warm_start': True, 'solver': 'sag', 'C': 4.9},
{'warm_start': True, 'solver': 'lbfgs', 'C': 3.9000000000000004},
{'warm_start': False, 'solver': 'newton-cg', 'C': 4.9},
{'warm_start': False, 'solver': 'newton-cg', 'C': 3.2},
{'warm_start': True, 'solver': 'newton-cg', 'C': 0.4},
{'warm_start': True, 'solver': 'newton-cg', 'C': 4.1},
{'warm_start': True, 'solver': 'sag', 'C': 3.0000000000000004},
{'warm_start': False, 'solver': 'lbfgs', 'C': 0.6},
{'warm_start': True, 'solver': 'lbfgs', 'C': 3.2},
{'warm_start': True, 'solver': 'sag', 'C': 4.2},
```

```
 {'warm_start': False, 'solver': 'sag', 'C': 1.6},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 1.4000000000000001},
 {'warm_start': True, 'solver': 'newton-cg', 'C': 0.1},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 0.30000000000000004},
 {'warm_start': True, 'solver': 'lbfgs', 'C': 4.3},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 1.2000000000000002},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 0.9},
 {'warm_start': True, 'solver': 'sag', 'C': 2.7},
 {'warm_start': False, 'solver': 'sag', 'C': 2.9000000000000004},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 4.7},
 {'warm_start': True, 'solver': 'lbfgs', 'C': 4.0},
 {'warm_start': False, 'solver': 'sag', 'C': 2.3000000000000003},
 {'warm_start': False, 'solver': 'sag', 'C': 4.0},
 {'warm_start': True, 'solver': 'newton-cg', 'C': 2.0},
 {'warm_start': True, 'solver': 'sag', 'C': 1.8000000000000003},
 {'warm_start': False, 'solver': 'lbfgs', 'C': 0.1},
 {'warm_start': True, 'solver': 'sag', 'C': 3.6},
 {'warm_start': True, 'solver': 'newton-cg', 'C': 4.9},
 {'warm_start': False, 'solver': 'lbfgs', 'C': 2.5000000000000004},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 3.4000000000000004},
 {'warm_start': False, 'solver': 'sag', 'C': 2.8000000000000003},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 3.3000000000000003},
 {'warm_start': True, 'solver': 'sag', 'C': 3.8000000000000003},
 {'warm_start': False, 'solver': 'sag', 'C': 3.1},
 {'warm_start': True, 'solver': 'newton-cg', 'C': 0.9},
 {'warm_start': False, 'solver': 'lbfgs', 'C': 3.2},
 {'warm_start': True, 'solver': 'sag', 'C': 1.1},
 {'warm_start': True, 'solver': 'newton-cg', 'C': 3.8000000000000003},
 {'warm_start': True, 'solver': 'sag', 'C': 3.5000000000000004},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 3.1},
 {'warm_start': True, 'solver': 'sag', 'C': 4.1},
 {'warm_start': True, 'solver': 'newton-cg', 'C': 4.8},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 0.2},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 3.7},
 {'warm_start': False, 'solver': 'sag', 'C': 3.4000000000000004},
 {'warm_start': False, 'solver': 'sag', 'C': 4.6},
 {'warm_start': True, 'solver': 'lbfgs', 'C': 4.9},
 {'warm_start': False, 'solver': 'sag', 'C': 4.2},
 {'warm_start': True, 'solver': 'lbfgs', 'C': 3.5000000000000004},
 {'warm_start': False, 'solver': 'lbfgs', 'C': 2.2},
 {'warm_start': False, 'solver': 'lbfgs', 'C': 3.1},
 {'warm_start': False, 'solver': 'sag', 'C': 1.1},
 {'warm_start': True, 'solver': 'lbfgs', 'C': 1.8000000000000003},
 {'warm_start': False, 'solver': 'newton-cg', 'C': 4.8},
 {'warm_start': False, 'solver': 'sag', 'C': 4.9}],
'split0_test_score': array([0.5       , 0.6650532 , 0.6444161 , 0.64314604, 0.5       ,
       0.6650532 , 0.5       , 0.6259103 , 0.5       , 0.5       ,
       0.64314604, 0.6650532 , 0.64074219, 0.5       , 0.6650532 ,
       0.5       , 0.63212432, 0.6650532 , 0.5       , 0.6650532 ,
       0.62845042, 0.62985675, 0.6650532 , 0.5       , 0.6650532 ,
       0.5       , 0.5       , 0.63212432, 0.5       , 0.6650532 ,
       0.5       , 0.62845042, 0.6650532 , 0.6650532 , 0.6650532 ,
       0.6650532 , 0.66732077, 0.6650532 , 0.5       , 0.64314604,
       0.66618699, 0.6338032 , 0.5       , 0.5       , 0.6650532 ,
       0.6650532 , 0.65303397, 0.6259103 , 0.5       , 0.62845042,
       0.62958421, 0.5       , 0.5       , 0.5       , 0.6650532 ,
       0.5       , 0.63212432, 0.5       , 0.66491693, 0.6650532 ,
       0.6650532 , 0.5       , 0.5       , 0.62845042, 0.5       ,
       0.6650532 , 0.64822628, 0.5       , 0.63212432, 0.6259103 ,
       0.64681995, 0.66732077, 0.6650532 , 0.6650532 , 0.63085426,
       0.66732077, 0.65403148, 0.6650532 , 0.6650532 , 0.6650532 ,
       0.64074219, 0.5       , 0.5       , 0.6650532 , 0.62718036,
       0.5       , 0.65770539, 0.5       , 0.5       , 0.62718036,
       0.5       , 0.5       , 0.64314604, 0.6650532 , 0.6650532 ,
       0.6650532 , 0.62845042, 0.6650532 , 0.64681995, 0.6650532 ,
       0.62958421, 0.6650532 , 0.5       , 0.62464024, 0.65784166,
       0.64201225, 0.5       , 0.5       , 0.5       , 0.5       ,
       0.62958421, 0.5       , 0.64201225, 0.5       , 0.5       ,
       0.5       , 0.5       , 0.64681995, 0.62845042, 0.6650532 ,
       0.66618699, 0.5       , 0.6650532 , 0.66732077, 0.6650532 ,
```

```
       0.62845042, 0.5       , 0.63085426, 0.62985675, 0.6650532 ,
       0.65190018, 0.5       , 0.65190018, 0.64681995, 0.5       ,
       0.63212432, 0.6650532 , 0.64201225, 0.66732077, 0.6650532 ,
       0.5       , 0.62845042, 0.65190018, 0.6650532 , 0.64554989,
       0.64681995, 0.65770539, 0.6650532 , 0.64314604, 0.6650532 ,
       0.62845042, 0.5       , 0.5       , 0.6650532 , 0.6650532 ,
       0.5       , 0.6650532 , 0.65303397, 0.6650532 , 0.5       ,
       0.5       , 0.5       , 0.5       , 0.5       , 0.5       ,
       0.6650532 , 0.6650532 , 0.6650532 , 0.6650532 , 0.64074219,
       0.6650532 , 0.6650532 , 0.65190018, 0.6444161 , 0.64074219,
       0.62972048, 0.63947214, 0.6650532 , 0.6650532 , 0.6650532 ,
       0.5       , 0.5       , 0.6259103 , 0.64201225, 0.63212432,
       0.64427983, 0.62972048, 0.64455237, 0.64681995, 0.62958421,
       0.63239687, 0.5       , 0.5       , 0.64074219, 0.5       ,
       0.5       , 0.63466444, 0.6650532 , 0.6650532 , 0.66491693,
       0.6650532 , 0.5       , 0.62718036, 0.62845042, 0.5       ,
       0.5       , 0.6650532 , 0.63947214, 0.66124302, 0.64681995,
       0.6650532 , 0.66618699, 0.5       , 0.5       , 0.6650532 ,
       0.65784166, 0.5       , 0.5       , 0.6650532 , 0.5       ,
       0.62464024, 0.5       , 0.6650532 , 0.64427983, 0.6650532 ,
       0.5       , 0.6650532 , 0.5       , 0.5       , 0.66618699,
       0.62845042, 0.5       , 0.6650532 , 0.5       , 0.6650532 ,
       0.5       , 0.6650532 , 0.65403148, 0.6650532 , 0.5       ,
       0.5       , 0.64201225, 0.5       , 0.62845042, 0.64455237,
       0.64681995, 0.5       , 0.6338032 , 0.6650532 , 0.5       ]),
'split1_test_score': array([0.5       , 0.62302632, 0.60400718, 0.60729665, 0.5       ,
       0.62188995, 0.5       , 0.60705742, 0.5       , 0.5       ,
       0.60161483, 0.62302632, 0.60047847, 0.5       , 0.62188995,
       0.5       , 0.59922249, 0.62302632, 0.5       , 0.62302632,
       0.60992823, 0.61937799, 0.62302632, 0.5       , 0.62188995,
       0.5       , 0.5       , 0.59922249, 0.5       , 0.62188995,
       0.5       , 0.59808612, 0.62416268, 0.62416268, 0.62302632,
       0.62302632, 0.6229067 , 0.62188995, 0.5       , 0.60161483,
       0.6229067 , 0.59934211, 0.5       , 0.5       , 0.62416268,
       0.62302632, 0.59946172, 0.59545455, 0.5       , 0.59808612,
       0.59683014, 0.5       , 0.5       , 0.5       , 0.62302632,
       0.5       , 0.59922249, 0.5       , 0.62278708, 0.62302632,
       0.62188995, 0.5       , 0.5       , 0.59694976, 0.5       ,
       0.62302632, 0.60047847, 0.5       , 0.59922249, 0.59545455,
       0.60047847, 0.62404306, 0.62302632, 0.62302632, 0.59922249,
       0.62278708, 0.61549043, 0.62302632, 0.62302632, 0.62188995,
       0.60023923, 0.5       , 0.5       , 0.62302632, 0.59796651,
       0.5       , 0.59922249, 0.5       , 0.5       , 0.60388756,
       0.5       , 0.5       , 0.60729665, 0.62302632, 0.62302632,
       0.62188995, 0.59808612, 0.62302632, 0.60047847, 0.62302632,
       0.60047847, 0.62302632, 0.5       , 0.60101675, 0.60047847,
       0.59808612, 0.5       , 0.5       , 0.5       , 0.5       ,
       0.59683014, 0.5       , 0.60047847, 0.5       , 0.5       ,
       0.5       , 0.5       , 0.60047847, 0.60047847, 0.62302632,
       0.6229067 , 0.5       , 0.62302632, 0.62278708, 0.62302632,
       0.60047847, 0.5       , 0.59922249, 0.61937799, 0.62302632,
       0.59922249, 0.5       , 0.61208134, 0.60047847, 0.5       ,
       0.59808612, 0.62302632, 0.59808612, 0.62404306, 0.62188995,
       0.5       , 0.62063397, 0.61208134, 0.62302632, 0.59922249,
       0.59808612, 0.59922249, 0.62302632, 0.61710526, 0.62188995,
       0.59694976, 0.5       , 0.5       , 0.62188995, 0.62302632,
       0.5       , 0.62302632, 0.59946172, 0.62188995, 0.5       ,
       0.5       , 0.5       , 0.5       , 0.5       , 0.5       ,
       0.62302632, 0.62302632, 0.62188995, 0.62302632, 0.61561005,
       0.62302632, 0.62302632, 0.59922249, 0.60400718, 0.60047847,
       0.59946172, 0.61297847, 0.62188995, 0.62302632, 0.62302632,
       0.5       , 0.5       , 0.60705742, 0.60047847, 0.59808612,
       0.6027512 , 0.59946172, 0.59922249, 0.60161483, 0.60047847,
       0.60047847, 0.5       , 0.5       , 0.60023923, 0.5       ,
       0.5       , 0.59922249, 0.62188995, 0.62302632, 0.62278708,
       0.62188995, 0.5       , 0.59796651, 0.59808612, 0.5       ,
       0.5       , 0.62302632, 0.61297847, 0.6215311 , 0.60161483,
       0.62302632, 0.62416268, 0.5       , 0.5       , 0.62188995,
       0.60047847, 0.5       , 0.5       , 0.62302632, 0.5       ,
       0.60101675, 0.5       , 0.62188995, 0.6027512 , 0.62302632,
```

```
       0.5       , 0.62302632, 0.5       , 0.5       , 0.62416268,
       0.59808612, 0.5       , 0.62188995, 0.5       , 0.62302632,
       0.5       , 0.62188995, 0.61549043, 0.62302632, 0.5       ,
       0.5       , 0.60047847, 0.5       , 0.59808612, 0.59922249,
       0.59922249, 0.5       , 0.59934211, 0.62188995, 0.5       ]),
'split2_test_score': array([0.5       , 0.67229021, 0.68037587, 0.66835664, 0.5       ,
       0.66988636, 0.5       , 0.66923077, 0.5       , 0.5       ,
       0.67089161, 0.67342657, 0.66354895, 0.5       , 0.66988636,
       0.5       , 0.66988636, 0.66988636, 0.5       , 0.67229021,
       0.67089161, 0.67810315, 0.67229021, 0.5       , 0.66988636,
       0.5       , 0.5       , 0.66254371, 0.5       , 0.66988636,
       0.5       , 0.6611451 , 0.67342657, 0.67342657, 0.66988636,
       0.67342657, 0.67342657, 0.66988636, 0.5       , 0.67089161,
       0.67342657, 0.66975524, 0.5       , 0.5       , 0.67342657,
       0.67342657, 0.66975524, 0.65620629, 0.5       , 0.6611451 ,
       0.66962413, 0.5       , 0.5       , 0.5       , 0.67229021,
       0.5       , 0.66988636, 0.5       , 0.67329545, 0.67229021,
       0.66988636, 0.5       , 0.5       , 0.67076049, 0.5       ,
       0.67342657, 0.67089161, 0.5       , 0.66254371, 0.65620629,
       0.66127622, 0.67342657, 0.67342657, 0.67342657, 0.6673514 ,
       0.67342657, 0.67076049, 0.67342657, 0.66988636, 0.66988636,
       0.66582168, 0.5       , 0.5       , 0.67342657, 0.6611451 ,
       0.5       , 0.66481643, 0.5       , 0.5       , 0.65659965,
       0.5       , 0.5       , 0.66835664, 0.67229021, 0.67229021,
       0.66988636, 0.67089161, 0.67342657, 0.66127622, 0.67229021,
       0.6743007 , 0.67342657, 0.5       , 0.64392483, 0.66975524,
       0.66468531, 0.5       , 0.5       , 0.5       , 0.5       ,
       0.66962413, 0.5       , 0.6673514 , 0.5       , 0.5       ,
       0.5       , 0.5       , 0.66975524, 0.67089161, 0.67229021,
       0.67342657, 0.5       , 0.67342657, 0.67342657, 0.67342657,
       0.67089161, 0.5       , 0.6673514 , 0.67810315, 0.66988636,
       0.66507867, 0.5       , 0.66988636, 0.66975524, 0.5       ,
       0.66975524, 0.67229021, 0.66468531, 0.67342657, 0.66988636,
       0.5       , 0.66494755, 0.66988636, 0.67342657, 0.6673514 ,
       0.67189685, 0.66481643, 0.67229021, 0.6673514 , 0.66988636,
       0.67076049, 0.5       , 0.5       , 0.66988636, 0.67342657,
       0.5       , 0.67342657, 0.66975524, 0.66988636, 0.5       ,
       0.5       , 0.5       , 0.5       , 0.5       , 0.5       ,
       0.66988636, 0.67229021, 0.66988636, 0.67342657, 0.67189685,
       0.67229021, 0.67342657, 0.66507867, 0.68037587, 0.66354895,
       0.66848776, 0.65253497, 0.66988636, 0.67229021, 0.67342657,
       0.5       , 0.5       , 0.66923077, 0.6673514 , 0.66975524,
       0.66975524, 0.66848776, 0.6673514 , 0.67329545, 0.6743007 ,
       0.66341783, 0.5       , 0.5       , 0.66582168, 0.5       ,
       0.5       , 0.66722028, 0.66988636, 0.67229021, 0.67329545,
       0.66988636, 0.5       , 0.6611451 , 0.67089161, 0.5       ,
       0.5       , 0.67342657, 0.65253497, 0.67443182, 0.67329545,
       0.67342657, 0.67102273, 0.5       , 0.5       , 0.66988636,
       0.66975524, 0.5       , 0.5       , 0.67342657, 0.5       ,
       0.64392483, 0.5       , 0.66988636, 0.66975524, 0.67229021,
       0.5       , 0.67229021, 0.5       , 0.5       , 0.67102273,
       0.67089161, 0.5       , 0.66988636, 0.5       , 0.67229021,
       0.5       , 0.66988636, 0.67076049, 0.66988636, 0.5       ,
       0.5       , 0.67229021, 0.5       , 0.67089161, 0.6673514 ,
       0.6673514 , 0.5       , 0.66975524, 0.66988636, 0.5       ]),
'mean_test_score': array([0.5       , 0.65345658, 0.64293305, 0.63959978, 0.5       ,
       0.65227651, 0.5       , 0.63406616, 0.5       , 0.5       ,
       0.63855083, 0.65383536, 0.6349232 , 0.5       , 0.65227651,
       0.5       , 0.63374439, 0.65265529, 0.5       , 0.65345658,
       0.63642342, 0.64244596, 0.65345658, 0.5       , 0.65227651,
       0.5       , 0.5       , 0.63129684, 0.5       , 0.65227651,
       0.5       , 0.62922722, 0.65421415, 0.65421415, 0.65265529,
       0.65383536, 0.65455135, 0.65227651, 0.5       , 0.63855083,
       0.65417342, 0.63430018, 0.5       , 0.5       , 0.65421415,
       0.65383536, 0.64075031, 0.62585705, 0.5       , 0.62922722,
       0.63201282, 0.5       , 0.5       , 0.5       , 0.65345658,
       0.5       , 0.63374439, 0.5       , 0.65366649, 0.65345658,
       0.65227651, 0.5       , 0.5       , 0.63205356, 0.5       ,
       0.65383536, 0.63986545, 0.5       , 0.63129684, 0.62585705,
       0.63619155, 0.65493014, 0.65383536, 0.65383536, 0.63247605,
```

```
       0.65451148, 0.6467608 , 0.65383536, 0.65265529, 0.65227651,
       0.63560104, 0.5       , 0.5       , 0.65383536, 0.62876399,
       0.5       , 0.64058144, 0.5       , 0.5       , 0.62922252,
       0.5       , 0.5       , 0.63959978, 0.65345658, 0.65345658,
       0.65227651, 0.63247605, 0.65383536, 0.63619155, 0.65345658,
       0.63478779, 0.65383536, 0.5       , 0.62319394, 0.64269179,
       0.6349279 , 0.5       , 0.5       , 0.5       , 0.5       ,
       0.63201282, 0.5       , 0.63661404, 0.5       , 0.5       ,
       0.5       , 0.5       , 0.63901789, 0.6332735 , 0.65345658,
       0.65417342, 0.5       , 0.65383536, 0.65451148, 0.65383536,
       0.6332735 , 0.5       , 0.63247605, 0.64244596, 0.65265529,
       0.63873378, 0.5       , 0.64462263, 0.63901789, 0.5       ,
       0.6333219 , 0.65345658, 0.6349279 , 0.65493014, 0.65227651,
       0.5       , 0.63801065, 0.64462263, 0.65383536, 0.63737459,
       0.63893431, 0.64058144, 0.65345658, 0.64253423, 0.65227651,
       0.63205356, 0.5       , 0.5       , 0.65227651, 0.65383536,
       0.5       , 0.65383536, 0.64075031, 0.65227651, 0.5       ,
       0.5       , 0.5       , 0.5       , 0.5       , 0.5       ,
       0.65265529, 0.65345658, 0.65227651, 0.65383536, 0.6427497 ,
       0.65345658, 0.65383536, 0.63873378, 0.64293305, 0.6349232 ,
       0.63255665, 0.63499519, 0.65227651, 0.65345658, 0.65383536,
       0.5       , 0.5       , 0.63406616, 0.63661404, 0.6333219 ,
       0.63892876, 0.63255665, 0.63704209, 0.64057674, 0.63478779,
       0.63209772, 0.5       , 0.5       , 0.63560104, 0.5       ,
       0.5       , 0.6337024 , 0.65227651, 0.65345658, 0.65366649,
       0.65227651, 0.5       , 0.62876399, 0.63247605, 0.5       ,
       0.5       , 0.65383536, 0.63499519, 0.65240198, 0.64057674,
       0.65383536, 0.6537908 , 0.5       , 0.5       , 0.65227651,
       0.64269179, 0.5       , 0.5       , 0.65383536, 0.5       ,
       0.62319394, 0.5       , 0.65227651, 0.63892876, 0.65345658,
       0.5       , 0.65345658, 0.5       , 0.5       , 0.6537908 ,
       0.63247605, 0.5       , 0.65227651, 0.5       , 0.65345658,
       0.5       , 0.65227651, 0.6467608 , 0.65265529, 0.5       ,
       0.5       , 0.63826031, 0.5       , 0.63247605, 0.63704209,
       0.63779794, 0.5       , 0.63430018, 0.65227651, 0.5       ]),
'std_test_score': array([0.        , 0.02171933, 0.03119502, 0.02505344, 0.        ,
       0.02157695, 0.        , 0.02602909, 0.        , 0.        ,
       0.02846817, 0.02205185, 0.02607511, 0.        , 0.02157695,
       0.        , 0.02887114, 0.02104356, 0.        , 0.02171933,
       0.02551875, 0.02557378, 0.02171933, 0.        , 0.02157695,
       0.        , 0.        , 0.0258574 , 0.        , 0.02157695,
       0.        , 0.02574958, 0.0215228 , 0.0215228 , 0.02104356,
       0.02205185, 0.02251456, 0.02157695, 0.        , 0.02846817,
       0.02230559, 0.02874819, 0.        , 0.        , 0.0215228 ,
       0.02205185, 0.0299829 , 0.02480183, 0.        , 0.02574958,
       0.0297676 , 0.        , 0.        , 0.        , 0.02171933,
       0.        , 0.02887114, 0.        , 0.02210133, 0.02171933,
       0.02157695, 0.        , 0.        , 0.03024062, 0.        ,
       0.02205185, 0.02934769, 0.        , 0.0258574 , 0.02480183,
       0.02593343, 0.02198225, 0.02205185, 0.02205185, 0.02783714,
       0.0225706 , 0.0231422 , 0.02205185, 0.02104356, 0.02157695,
       0.0270196 , 0.        , 0.        , 0.02205185, 0.02581685,
       0.        , 0.02938893, 0.        , 0.        , 0.02156802,
       0.        , 0.        , 0.02505344, 0.02171933, 0.02171933,
       0.02157695, 0.02985871, 0.02205185, 0.02593343, 0.02171933,
       0.03036158, 0.02205185, 0.        , 0.01754698, 0.03024298,
       0.02764663, 0.        , 0.        , 0.        , 0.        ,
       0.0297676 , 0.        , 0.02756632, 0.        , 0.        ,
       0.        , 0.        , 0.02881518, 0.02894764, 0.02171933,
       0.02230559, 0.        , 0.02205185, 0.0225706 , 0.02205185,
       0.02894764, 0.        , 0.02783714, 0.02557378, 0.02104356,
       0.028452  , 0.        , 0.02415336, 0.02881518, 0.        ,
       0.02927105, 0.02171933, 0.02764663, 0.02198225, 0.02157695,
       0.        , 0.01931272, 0.02415336, 0.02205185, 0.02840791,
       0.03064467, 0.02938893, 0.02171933, 0.02051746, 0.02157695,
       0.03024062, 0.        , 0.        , 0.02157695, 0.02205185,
       0.        , 0.02205185, 0.0299829 , 0.02157695, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.02104356, 0.02171933, 0.02157695, 0.02205185, 0.0230228 ,
       0.02171933, 0.02205185, 0.028452  , 0.03119502, 0.02607511,
```

```
       0.02825103, 0.01645623, 0.02157695, 0.02171933, 0.02205185,
       0.        , 0.        , 0.02602909, 0.02756632, 0.02927105,
       0.02761474, 0.02825103, 0.02831596, 0.02959461, 0.03036158,
       0.02569576, 0.        , 0.        , 0.0270196 , 0.        ,
       0.        , 0.02776832, 0.02157695, 0.02171933, 0.02210133,
       0.02157695, 0.        , 0.02581685, 0.02985871, 0.        ,
       0.        , 0.02205185, 0.01645623, 0.02248325, 0.02959461,
       0.02205185, 0.02104305, 0.        , 0.        , 0.02157695,
       0.03024298, 0.        , 0.        , 0.02205185, 0.        ,
       0.01754698, 0.        , 0.02157695, 0.02761474, 0.02171933,
       0.        , 0.02171933, 0.        , 0.        , 0.02104305,
       0.02985871, 0.        , 0.02157695, 0.        , 0.02171933,
       0.        , 0.02157695, 0.0231422 , 0.02104356, 0.        ,
       0.        , 0.02943682, 0.        , 0.02985871, 0.02831596,
       0.02853576, 0.        , 0.02874819, 0.02157695, 0.        ]),
'rank_test_score': array([168,  36,  86, 101, 168,  61, 168, 135, 168, 168, 110,  11, 129,
       168,  61, 168, 137,  54, 168,  36, 120,  92,  36, 168,  61, 168,
       168, 157, 168,  61, 168, 159,   6,   6,  54,  11,   3,  61, 168,
       110,   9, 133, 168, 168,   6,  11,  94, 164, 168, 159, 155, 168,
       168, 168,  36, 168, 137, 168,  34,  36,  61, 168, 168, 153, 168,
        11, 100, 168, 157, 164, 121,   1,  11,  11, 146,   4,  82,  11,
        54,  61, 123, 168, 168,  11, 162, 168,  96, 168, 168, 161, 168,
       168, 101,  36,  36,  61, 146,  11, 121,  36, 131,  11, 168, 166,
        89, 127, 168, 168, 168, 168, 155, 168, 118, 168, 168, 168, 168,
       103, 142,  36,   9, 168,  11,   4,  11, 142, 168, 146,  92,  54,
       108, 168,  84, 103, 168, 140,  36, 127,   1,  61, 168, 113,  84,
        11, 115, 105,  96,  36,  91,  61, 153, 168, 168,  61,  11, 168,
        11,  94,  61, 168, 168, 168, 168, 168, 168,  54,  36,  61,  11,
        88,  36,  11, 108,  86, 129, 144, 125,  61,  36,  11, 168, 168,
       135, 118, 140, 106, 144, 116,  98, 131, 152, 168, 168, 123, 168,
       168, 139,  61,  36,  34,  61, 168, 162, 146, 168, 168,  11, 125,
        60,  98,  11,  32, 168, 168,  61,  89, 168, 168,  11, 168, 166,
       168,  61, 106,  36, 168,  36, 168, 168,  32, 146, 168,  61, 168,
        36, 168,  61,  82,  54, 168, 168, 112, 168, 146, 116, 114, 168,
       133,  61, 168], dtype=int32)}
```

In [60…

```
# checking the best estimator for the model
lr_tuned_cv.best_estimator_
```

Out[601]:

```
LogisticRegression(C=0.6, max_iter=1000, random_state=219, solver='newton-cg',
                   warm_start=True)
```

In [60…

```
# Tuned LR
```

In [60…

```
# building a model based on hyperparameter tuning results

# INSTANTIATING a logistic regression model with tuned values
lr_tuned = lr_tuned_cv.best_estimator_


# FIT step is not needed


# PREDICTING based on the testing set
lr_tuned_pred = lr_tuned.predict(X_test)


# SCORING the results
print('LR Tuned Training ACCURACY:', lr_tuned.score(X_train, y_train).round(4))
print('LR Tuned Testing  ACCURACY:', lr_tuned.score(X_test, y_test).round(4))
print('LR Tuned AUC Score        :', roc_auc_score(y_true  = y_test,
                                                   y_score = lr_tuned_pred).round(4))


# saving scoring data for future use
```

```
lr_tuned_train_score = lr_tuned.score(X_train, y_train).round(4) # accuracy
lr_tuned_test_score  = lr_tuned.score(X_test, y_test).round(4)   # accuracy


# saving the AUC score
lr_tuned_auc         = roc_auc_score(y_true  = y_test,
                                     y_score = lr_tuned_pred).round(4) # auc
```

```
LR Tuned Training ACCURACY: 0.7443
LR Tuned Testing  ACCURACY: 0.7474
LR Tuned AUC Score        : 0.6549
```

In [60…

```
# unpacking the confusion matrix
lr_tuned_tn, \
lr_tuned_fp, \
lr_tuned_fn, \
lr_tuned_tp = confusion_matrix(y_true = y_test, y_pred = lr_tuned_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {lr_tuned_tn}
False Positives: {lr_tuned_fp}
False Negatives: {lr_tuned_fn}
True Positives : {lr_tuned_tp}
""")
```

```
True Negatives : 62
False Positives: 94
False Negatives: 29
True Positives : 302
```

In [60…

```
# loading model performance

# declaring model performance objects
lr_train_acc = lr_tuned.score(X_train, y_train).round(4)
lr_test_acc  = lr_tuned.score(X_test, y_test).round(4)
lr_auc       = roc_auc_score(y_true  = y_test,
                             y_score = lr_tuned_pred).round(4)


# appending to model_performance
model_performance = model_performance.append(
                         {'Model Name'        : 'Tuned LR',
                          'Training Accuracy' : lr_train_acc,
                          'Testing Accuracy'  : lr_test_acc,
                          'AUC Score'         : lr_auc,
                          'Confusion Matrix'  : (lr_tuned_tn,
                                                 lr_tuned_fp,
                                                 lr_tuned_fn,
                                                 lr_tuned_tp)},
                          ignore_index = True)


# checking the results
model_performance
```

Out[607]:

	Model Name	AUC Score	Training Accuracy	Testing Accuracy	Confusion Matrix
0	Logistic	0.6132	0.7231	0.7207	(49, 107, 29, 302)
1	Full Tree	0.6192	1.0000	0.6735	(73, 83, 76, 255)

	Model Name	AUC Score	Training Accuracy	Testing Accuracy	Confusion Matrix
2	Pruned Tree	0.7130	0.7512	0.7803	(82, 74, 33, 298)
3	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
4	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
5	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
6	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)

In [60…

```
# Tuned Tree
```

In [60…

```
# declaring a hyperparameter space
criterion_space = ['gini', 'entropy']
splitter_space  = ['best', 'random']
depth_space     = pd.np.arange(1, 25, 1)
leaf_space      = pd.np.arange(1, 100, 1)


# creating a hyperparameter grid
param_grid = {'criterion'        : criterion_space,
              'splitter'         : splitter_space,
              'max_depth'        : depth_space,
              'min_samples_leaf' : leaf_space}


# INSTANTIATING the model object without hyperparameters
tuned_tree = DecisionTreeClassifier(random_state = 219)


# RandomizedSearchCV object
tuned_tree_cv = RandomizedSearchCV(estimator             = tuned_tree,
                                   param_distributions   = param_grid,
                                   cv                    = 3,
                                   n_iter                = 1000,
                                   random_state          = 219,
                                   scoring = make_scorer(roc_auc_score,
                                             needs_threshold = False))


# FITTING to the FULL DATASET (due to cross-validation)
tuned_tree_cv.fit(chef_data, chef_target)


# PREDICT step is not needed


# printing the optimal parameters and best score
print("Tuned Parameters  :", tuned_tree_cv.best_params_)
print("Tuned Training AUC:", tuned_tree_cv.best_score_.round(4))
```

```
<ipython-input-609-4dc3f988466f>:4: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  depth_space     = pd.np.arange(1, 25, 1)
<ipython-input-609-4dc3f988466f>:5: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  leaf_space      = pd.np.arange(1, 100, 1)
Tuned Parameters  : {'splitter': 'best', 'min_samples_leaf': 16, 'max_depth': 3, 'criterion': 'gini'}
Tuned Training AUC: 0.7032
```

In [61…

```
# building a model based on hyperparameter tuning results

# INSTANTIATING a logistic regression model with tuned values
```

```
tree_tuned = tuned_tree_cv.best_estimator_


# FIT step is not needed


# PREDICTING based on the testing set
tree_tuned_pred = tree_tuned.predict(X_test)


# SCORING the results
print('Training ACCURACY:', tree_tuned.score(X_train, y_train).round(4))
print('Testing  ACCURACY:', tree_tuned.score(X_test, y_test).round(4))
print('AUC Score        :', roc_auc_score(y_true  = y_test,
                                          y_score = tree_tuned_pred).round(4))


# saving scoring data for future use
tree_tuned_train_score = tree_tuned.score(X_train, y_train).round(4) # accuracy
tree_tuned_test_score  = tree_tuned.score(X_test, y_test).round(4)   # accuracy


# saving the AUC score
tree_tuned_auc         = roc_auc_score(y_true  = y_test,
                                       y_score = tree_tuned_pred).round(4) # auc
```

```
Training ACCURACY: 0.7402
Testing  ACCURACY: 0.7762
AUC Score        : 0.732
```

In [61…

```
# unpacking the confusion matrix
tuned_tree_tn, \
tuned_tree_fp, \
tuned_tree_fn, \
tuned_tree_tp = confusion_matrix(y_true = y_test, y_pred = tree_tuned_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {tuned_tree_tn}
False Positives: {tuned_tree_fp}
False Negatives: {tuned_tree_fn}
True Positives : {tuned_tree_tp}
""")
```

```
True Negatives : 95
False Positives: 61
False Negatives: 48
True Positives : 283
```

In [61…

```
# declaring model performance objects
tree_train_acc = tree_tuned.score(X_train, y_train).round(4)
tree_test_acc  = tree_tuned.score(X_test, y_test).round(4)
tree_auc       = roc_auc_score(y_true  = y_test,
                               y_score = tree_tuned_pred).round(4)


# appending to model_performance
model_performance = model_performance.append(
                         {'Model Name'         : 'Tuned Tree',
                          'Training Accuracy'  : tree_train_acc,
                          'Testing Accuracy'   : tree_test_acc,
                          'AUC Score'          : tree_auc,
                          'Confusion Matrix'   : (tuned_tree_tn,
```

```
                                            tuned_tree_fp,
                                            tuned_tree_fn,
                                            tuned_tree_tp)},
                           ignore_index = True)


# checking the results
model_performance
```

Out[612]:

	Model Name	AUC Score	Training Accuracy	Testing Accuracy	Confusion Matrix
0	Logistic	0.6132	0.7231	0.7207	(49, 107, 29, 302)
1	Full Tree	0.6192	1.0000	0.6735	(73, 83, 76, 255)
2	Pruned Tree	0.7130	0.7512	0.7803	(82, 74, 33, 298)
3	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
4	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
5	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
6	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
7	Tuned Tree	0.7320	0.7402	0.7762	(95, 61, 48, 283)

In [61…

```
#Ensemble Modeling
```

In [61…

```
########################################
# plot_feature_importances
########################################
def plot_feature_importances(model, train, export = False):
    """
    Plots the importance of features from a CART model.

    PARAMETERS
    ----------
    model  : CART model
    train  : explanatory variable training data
    export : whether or not to export as a .png image, default False
    """

    # declaring the number
    n_features = train.shape[1]

    # setting plot window
    fig, ax = plt.subplots(figsize=(12,9))

    plt.barh(range(n_features), model.feature_importances_, align='center')
    plt.yticks(pd.np.arange(n_features), train.columns)
    plt.xlabel("Feature importance")
    plt.ylabel("Feature")

    if export == True:
        plt.savefig('./analysis_images/Feature_Importance.png')
```

In [62…

```
# train/test split with the logit_sig variables
chef_data   =  chef.loc[ : , candidate_dict['logit_full']]
chef_target =  chef.loc[ : , 'CROSS_SELL_SUCCESS']


# train/test split
```

```
x_train, x_test, y_train, y_test = train_test_split(
            chef_data,
            chef_target,
            random_state = 219,
            test_size    = 0.25,
            stratify     = chef_target)
```

In [62…

```
#Random Forest (Full)
```

In [62…

```
# INSTANTIATING a random forest model with default values
rf_default = RandomForestClassifier(n_estimators     = 100,
                                    criterion        = 'gini',
                                    max_depth        = None,
                                    min_samples_leaf = 1,
                                    bootstrap        = True,
                                    warm_start       = False,
                                    random_state     = 219)
```

In [62…

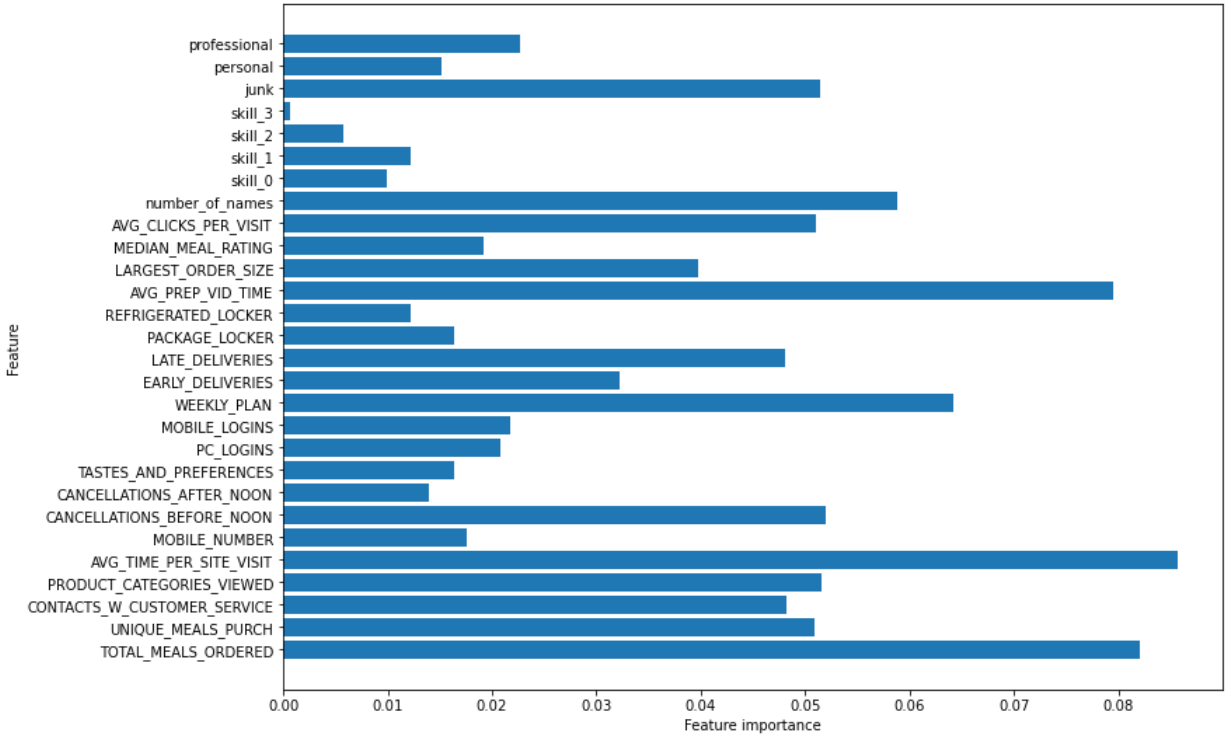
```
# FITTING the training data
rf_default_fit = rf_default.fit(x_train, y_train)


# PREDICTING based on the testing set
rf_default_fit_pred = rf_default_fit.predict(x_test)


# SCORING the results
print('Training ACCURACY:', rf_default_fit.score(x_train, y_train).round(4))
print('Testing  ACCURACY:', rf_default_fit.score(x_test, y_test).round(4))


# saving AUC score
print('AUC Score        :', roc_auc_score(y_true  = y_test,
                                          y_score = rf_default_fit_pred).round(4))
```

```
Training ACCURACY: 1.0
Testing  ACCURACY: 0.7556
AUC Score        : 0.6711
```

In [62…

```
# plotting feature importances
plot_feature_importances(rf_default_fit,
                         train = x_train,
                         export = False)
```

```
<ipython-input-614-141a9830ad1b>:22: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  plt.yticks(pd.np.arange(n_features), train.columns)
```



In [62…

```
# unpacking the confusion matrix
rf_tn, \
rf_fp, \
rf_fn, \
rf_tp = confusion_matrix(y_true = y_test, y_pred = rf_default_fit_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {rf_tn}
False Positives: {rf_fp}
False Negatives: {rf_fn}
True Positives : {rf_tp}
""")
```

```
True Negatives : 68
False Positives: 88
False Negatives: 31
True Positives : 300
```

In [62…

```
# declaring model performance objects
rf_train_acc = rf_default_fit.score(x_train, y_train).round(4)
rf_test_acc  = rf_default_fit.score(x_test, y_test).round(4)
rf_auc       = roc_auc_score(y_true  = y_test,
                             y_score = rf_default_fit_pred).round(4)


# appending to model_performance
model_performance = model_performance.append(
```

```
                        {'Model Name'          : 'Random Forest (Full)',
                         'Training Accuracy'   : rf_train_acc,
                         'Testing Accuracy'    : rf_test_acc,
                         'AUC Score'           : rf_auc,
                         'Confusion Matrix'    : (rf_tn,
                                                  rf_fp,
                                                  rf_fn,
                                                  rf_tp)},
                        ignore_index = True)


# checking the results
model_performance
```

Out[627]:

	Model Name	AUC Score	Training Accuracy	Testing Accuracy	Confusion Matrix
0	Logistic	0.6132	0.7231	0.7207	(49, 107, 29, 302)
1	Full Tree	0.6192	1.0000	0.6735	(73, 83, 76, 255)
2	Pruned Tree	0.7130	0.7512	0.7803	(82, 74, 33, 298)
3	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
4	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
5	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
6	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
7	Tuned Tree	0.7320	0.7402	0.7762	(95, 61, 48, 283)
8	Random Forest (Full)	0.6711	1.0000	0.7556	(68, 88, 31, 300)

In [62…

```
#Tuned Random Forest (Full)
```

In [ ]:

```
# FITTING the training data
rf_default_fit = rf_default.fit(x_train, y_train)


# PREDICTING based on the testing set
rf_default_fit_pred = rf_default_fit.predict(x_test)


# declaring a hyperparameter space
estimator_space  = pd.np.arange(100, 1100, 250)
leaf_space       = pd.np.arange(1, 31, 10)
criterion_space  = ['gini', 'entropy']
bootstrap_space  = [True, False]
warm_start_space = [True, False]


# creating a hyperparameter grid
param_grid = {'n_estimators'     : estimator_space,
              'min_samples_leaf' : leaf_space,
              'criterion'        : criterion_space,
              'bootstrap'        : bootstrap_space,
              'warm_start'       : warm_start_space}


# INSTANTIATING the model object without hyperparameters
forest_grid = RandomForestClassifier(random_state = 219)
```

```
# GridSearchCV object
forest_cv = RandomizedSearchCV(estimator           = forest_grid,
                               param_distributions = param_grid,
                               cv         = 3,
                               n_iter     = 1000,
                               scoring    = make_scorer(roc_auc_score,
                                            needs_threshold = False))


# FITTING to the FULL DATASET (due to cross-validation)
forest_cv.fit(chef_data, chef_target)


# PREDICT step is not needed


# printing the optimal parameters and best score
print("Tuned Parameters  :", forest_cv.best_params_)
print("Tuned Training AUC:", forest_cv.best_score_.round(4))
```

In [ ]:

```
# best estimators based on RandomizedSearchCV
forest_cv.best_estimator_
```

In [ ]:

```
# building a model based on hyperparameter tuning results

# copy/pasting in the best_estimator_ results
# to avoid running another RandomizedSearch
forest_tuned = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=11, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=350,
                       n_jobs=None, oob_score=False, random_state=219,
                       verbose=0, warm_start=True)


# FITTING the model object
forest_tuned_fit = forest_tuned.fit(x_train, y_train)


# PREDICTING based on the testing set
forest_tuned_pred = forest_tuned_fit.predict(x_test)


# SCORING the results
print('Forest Tuned Training ACCURACY:', forest_tuned.score(x_train, y_train).round(4))
print('Forest Tuned Testing  ACCURACY:', forest_tuned.score(x_test, y_test).round(4))
print('Forest Tuned AUC Score        :', roc_auc_score(y_true  = y_test,
                                                       y_score = forest_tuned_pred).round(4))


# saving scoring data for future use
forest_tuned_train_score = forest_tuned.score(x_train, y_train).round(4) # accuracy
forest_tuned_test_score  = forest_tuned.score(x_test, y_test).round(4)   # accuracy


# saving the AUC score
forest_tuned_auc = roc_auc_score(y_true  = y_test,
                                 y_score = forest_tuned_pred).round(4) # auc
```

In [ ]:

```
# plotting feature importances
plot_feature_importances(forest_tuned_fit,
                         train = x_train,
                         export = False)
```

In [ ]:

```
# unpacking the confusion matrix
tuned_rf_tn, \
tuned_rf_fp, \
tuned_rf_fn, \
tuned_rf_tp = confusion_matrix(y_true = y_test, y_pred = forest_tuned_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {tuned_rf_tn}
False Positives: {tuned_rf_fp}
False Negatives: {tuned_rf_fn}
True Positives : {tuned_rf_tp}
""")
```

In [ ]:

```
# declaring model performance objects
tuned_rf_train_acc = forest_tuned_fit.score(x_train, y_train).round(4)
tuned_rf_test_acc  = forest_tuned_fit.score(x_test, y_test).round(4)
tuned_rf_auc       = roc_auc_score(y_true  = y_test,
                                   y_score = forest_tuned_pred).round(4)


# appending to model_performance
model_performance = model_performance.append(
                         {'Model Name'         : 'Tuned Random Forest (Full)',
                          'Training Accuracy'  : tuned_rf_train_acc,
                          'Testing Accuracy'   : tuned_rf_test_acc,
                          'AUC Score'          : tuned_rf_auc,
                          'Confusion Matrix'   : (tuned_rf_tn,
                                                  tuned_rf_fp,
                                                  tuned_rf_fn,
                                                  tuned_rf_tp)},
                         ignore_index = True)


# checking the results
model_performance
```

In [ ]:

```
#GBM (Full)
```

In [ ]:

```

```

In [63…

```
# INSTANTIATING the model object without hyperparameters
full_gbm_default = GradientBoostingClassifier(loss          = 'deviance',
                                              learning_rate = 0.1,
                                              n_estimators  = 100,
                                              criterion     = 'friedman_mse',
                                              max_depth     = 3,
                                              warm_start    = False,
                                              random_state  = 219)


# FIT step is needed as we are not using .best_estimator
full_gbm_default_fit = full_gbm_default.fit(x_train, y_train)
```

```
# PREDICTING based on the testing set
full_gbm_default_pred = full_gbm_default_fit.predict(x_test)


# SCORING the results
print('Training ACCURACY:', full_gbm_default_fit.score(x_train, y_train).round(4))
print('Testing ACCURACY :', full_gbm_default_fit.score(x_test, y_test).round(4))
print('AUC Score        :', roc_auc_score(y_true  = y_test,
                                          y_score = full_gbm_default_pred).round(4))
```

```
Training ACCURACY: 0.8417
Testing ACCURACY : 0.7515
AUC Score        : 0.6698
```

In [63…

```
# unpacking the confusion matrix
gbm_default_tn, \
gbm_default_fp, \
gbm_default_fn, \
gbm_default_tp = confusion_matrix(y_true = y_test, y_pred = full_gbm_default_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {gbm_default_tn}
False Positives: {gbm_default_fp}
False Negatives: {gbm_default_fn}
True Positives : {gbm_default_tp}
""")
```

```
True Negatives : 69
False Positives: 87
False Negatives: 34
True Positives : 297
```

In [63…

```
# declaring model performance objects
gbm_train_acc = full_gbm_default_fit.score(x_train, y_train).round(4)
gbm_test_acc  = full_gbm_default_fit.score(x_test, y_test).round(4)
gbm_auc       = roc_auc_score(y_true  = y_test,
                              y_score = full_gbm_default_pred).round(4)


# appending to model_performance
model_performance = model_performance.append(
                          {'Model Name'       : 'GBM (Full)',
                          'Training Accuracy' : gbm_train_acc,
                          'Testing Accuracy'  : gbm_test_acc,
                          'AUC Score'         : gbm_auc,
                          'Confusion Matrix'  : (gbm_default_tn,
                                                 gbm_default_fp,
                                                 gbm_default_fn,
                                                 gbm_default_tp)},
                          ignore_index = True)


# checking the results
model_performance
```

Out[638]:

	Model Name	AUC Score	Training Accuracy	Testing Accuracy	Confusion Matrix
0	Logistic	0.6132	0.7231	0.7207	(49, 107, 29, 302)

	Model Name	AUC Score	Training Accuracy	Testing Accuracy	Confusion Matrix
1	Full Tree	0.6192	1.0000	0.6735	(73, 83, 76, 255)
2	Pruned Tree	0.7130	0.7512	0.7803	(82, 74, 33, 298)
3	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
4	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
5	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
6	Tuned LR	0.6549	0.7443	0.7474	(62, 94, 29, 302)
7	Tuned Tree	0.7320	0.7402	0.7762	(95, 61, 48, 283)
8	Random Forest (Full)	0.6711	1.0000	0.7556	(68, 88, 31, 300)
9	Tuned Random Forest (Full)	0.6310	0.8033	0.7495	(47, 109, 13, 318)
10	GBM (Full)	0.6698	0.8417	0.7515	(69, 87, 34, 297)

In [63…

```
#Tuned GBM
```

In [ ]:

```
# declaring a hyperparameter space
learn_space        = pd.np.arange(0.1, 2.0, 0.2)
estimator_space    = pd.np.arange(100, 200, 25)
depth_space        = pd.np.arange(1, 20, 2)
warm_start_space   = [True, False]

# creating a hyperparameter grid
param_grid = {'learning_rate' : learn_space,
              'max_depth'     : depth_space,
              'n_estimators'  : estimator_space,
              'warm_start'     : warm_start_space}


# INSTANTIATING the model object without hyperparameters
full_gbm_grid = GradientBoostingClassifier(random_state = 219)


# GridSearchCV object
full_gbm_cv = RandomizedSearchCV(estimator     = full_gbm_grid,
                         param_distributions = param_grid,
                         cv                  = 3,
                         n_iter              = 500,
                         random_state        = 219,
                         scoring             = make_scorer(roc_auc_score,
                                               needs_threshold = False))


# FITTING to the FULL DATASET (due to cross-validation)
full_gbm_cv.fit(chef_data, chef_target)


# PREDICT step is not needed


# printing the optimal parameters and best score
print("Tuned Parameters  :", full_gbm_cv.best_params_)
print("Tuned Training AUC:", full_gbm_cv.best_score_.round(4))
```

```
<ipython-input-640-4231b2b02a87>:2: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  learn_space        = pd.np.arange(0.1, 2.0, 0.2)
```

```
<ipython-input-640-4231b2b02a87>:3: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  estimator_space    = pd.np.arange(100, 200, 25)
<ipython-input-640-4231b2b02a87>:4: FutureWarning: The pandas.np module is deprecated and will be removed from pandas in a future version. Import numpy directly instead
  depth_space        = pd.np.arange(1, 20, 2)
```

In [ ]:

```
# checking the best estimator for the model
full_gbm_cv.best_estimator_
```

In [ ]:

```
# INSTANTIATING the model object without hyperparameters

# I made several attempts to hyperparameter tuning
gbm_tuned = GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.7000000000000001, loss='deviance',
                           max_depth=1, max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=125,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=219, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=True)

# FIT step is needed as we are not using .best_estimator
gbm_tuned_fit = gbm_tuned.fit(x_train, y_train)


# PREDICTING based on the testing set
gbm_tuned_pred = gbm_tuned_fit.predict(x_test)


# SCORING the results
print('Training ACCURACY:', gbm_tuned_fit.score(x_train, y_train).round(4))
print('Testing  ACCURACY:', gbm_tuned_fit.score(x_test, y_test).round(4))
print('AUC Score        :', roc_auc_score(y_true  = y_test,
                                          y_score = gbm_tuned_pred).round(4))
```

In [ ]:

```
# unpacking the confusion matrix
gbm_tuned_tn, \
gbm_tuned_fp, \
gbm_tuned_fn, \
gbm_tuned_tp = confusion_matrix(y_true = y_test, y_pred = gbm_tuned_pred).ravel()


# printing each result one-by-one
print(f"""
True Negatives : {gbm_tuned_tn}
False Positives: {gbm_tuned_fp}
False Negatives: {gbm_tuned_fn}
True Positives : {gbm_tuned_tp}
""")
```

In [ ]:

```
# declaring model performance objects
gbm_train_acc = gbm_tuned_fit.score(x_train, y_train).round(4)
gbm_test_acc  = gbm_tuned_fit.score(x_test, y_test).round(4)
gbm_auc       = roc_auc_score(y_true  = y_test,
                              y_score = gbm_tuned_pred).round(4)


# appending to model_performance
model_performance = model_performance.append(
                          {'Model Name'        : 'Tuned GBM',
```

```
                          'Training Accuracy'   : gbm_train_acc,
                          'Testing Accuracy'    : gbm_test_acc,
                          'AUC Score'           : gbm_auc,
                          'Confusion Matrix'    : (gbm_tuned_tn,
                                                   gbm_tuned_fp,
                                                   gbm_tuned_fn,
                                                   gbm_tuned_tp)},
                          ignore_index = True)


# checking the results
model_performance
```

In [ ]:
```
# Final results
```

In [ ]:
```
model_performance.sort_values(by = 'AUC Score',
                              ascending = False)
```

In [ ]:
```
# saving the DataFrame to Excel
model_performance.to_excel('./model_results/classification_model_performance.xlsx',
                           index = False)
```

In [ ]:
```
# Conclusion
```

In [ ]:
```
# The best I can get is Pruned Tree     0.7130  0.7512  0.7803  (82, 74, 33, 298)
```

In [ ]:
```
# 0.7803         Testing Accuracy and lowest Flase positive/negative error
```