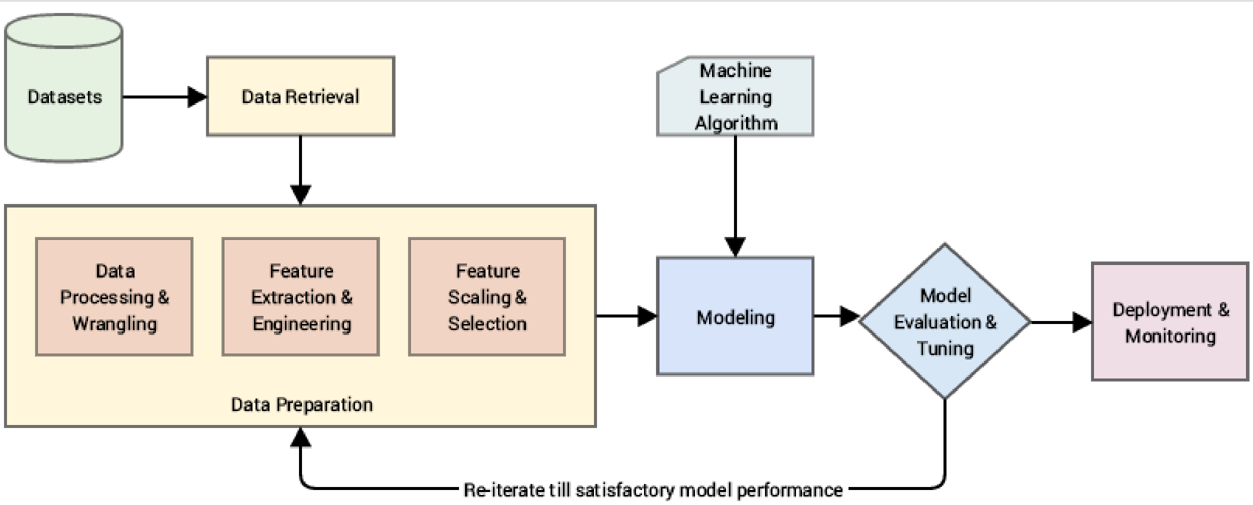
In [1]:
```
from IPython.core.interactiveshell import InteractiveShell   #Allowing for multiline ou
InteractiveShell.ast_node_interactivity = "all"
```

Importing Modules and Creating Workflow

In [2]:
```
import os
import time
from PIL import Image
import numpy as np
import pandas as pd
import pandas_flavor as pf
import matplotlib.pyplot as plt
import seaborn as sns
from statistics import mean
from statistics import stdev
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import scikitplot as skplt
```

In [3]:
```
img = Image.open('workflow.png')
display(img)
```



Data Exploration

cwd = os.getcwd() directory = os.fsencode(cwd) files = [] for file in os.listdir(directory): filename = os.fsdecode(file) if filename.endswith(".csv"): print(filename.split('.')[0])

In [4]:
```
#Reading in different data files and analyzing first few rows of each data set
training_df = pd.read_csv('FoodDelivery.csv') #Airport_codes -> ac
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [5]:
```
training_df.shape # We will split training data into training and validation
```
Out[5]:
```
(388, 50)
```

In [6]:
```
training_df.head()
```
Out[6]:

	Age	Gender	Marital_Status	Occupation	Monthly_Income	Educational_Qualifications	Family_size	la
0	NaN	Female	Single	Student	No Income	Post Graduate	4	1
1	24.0	Female	Single	Student	Below Rs.10000	Graduate	3	1
2	22.0	Male	Single	Student	Below Rs.10000	Post Graduate	3	1
3	22.0	Female	Single	Student	No Income	Graduate	6	1
4	22.0	Male	Single	Student	Below Rs.10000	Post Graduate	4	1

In [7]:
```
training_df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388 entries, 0 to 387
Data columns (total 50 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         379 non-null    float64
 1   Gender                      388 non-null    object
 2   Marital_Status              388 non-null    object
 3   Occupation                  388 non-null    object
 4   Monthly_Income              388 non-null    object
 5   Educational_Qualifications  388 non-null    object
 6   Family_size                 388 non-null    int64
 7   latitude                    388 non-null    float64
 8   longitude                   388 non-null    float64
 9   Meal(P1)                    388 non-null    object
 10  Meal(P2)                    388 non-null    object
 11  Perference(P1)              388 non-null    object
 12  Perference(P2)              388 non-null    object
 13  Ease_and_convenient         388 non-null    object
 14  Time_saving                 388 non-null    object
 15  More_restaurant_choices     388 non-null    object
 16  Easy_Payment_option         388 non-null    object
 17  More_Offers_and_Discount    388 non-null    object
 18  Good_Food_quality           388 non-null    object
 19  Good_Tracking_system        388 non-null    object
 20  Self_Cooking                388 non-null    object
 21  Health_Concern              388 non-null    object
 22  Late_Delivery               388 non-null    object
 23  Poor_Hygiene                388 non-null    object
 24  Bad_past_experience         388 non-null    object
```

```
 25  Unavailability                             388 non-null    object
 26  Unaffordable                               388 non-null    object
 27  Long_delivery_time                         388 non-null    object
 28  Delay_of_delivery_person_getting_assigned  388 non-null    object
 29  Delay_of_delivery_person_picking_up_food   388 non-null    object
 30  Wrong_order_delivered                      388 non-null    object
 31  Missing_item                               388 non-null    object
 32  Order_placed_by_mistake                    388 non-null    object
 33  Influence_of_time                          388 non-null    object
 34  Order_Time                                 388 non-null    object
 35  Maximum_wait_time                          388 non-null    object
 36  Residence_in_busy_location                 388 non-null    object
 37  Good_Road_Condition                        388 non-null    object
 38  Low_quantity_low_time                      388 non-null    object
 39  Delivery_person_ability                    388 non-null    object
 40  Influence_of_rating                        388 non-null    object
 41  Less_Delivery_time                         388 non-null    object
 42  High_Quality_of_package                    388 non-null    object
 43  Number_of_calls                            388 non-null    object
 44  Politeness                                 388 non-null    object
 45  Freshness                                  388 non-null    object
 46  Temperature                                388 non-null    object
 47  Good_Taste                                 388 non-null    object
 48  Good_Quantity                              388 non-null    object
 49  orderAgain                                 388 non-null    object
dtypes: float64(3), int64(1), object(46)
memory usage: 151.7+ KB
```

In [8]:

```
def missing_values_table(df):
    mis_val = df.isnull().sum()
    mis_val_percent = 100 * df.isnull().sum() / len(df)
    mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
    mis_val_table_ren_columns = mis_val_table.rename(
    columns = {0 : 'Missing Values', 1 : '% of Total Values'})
    mis_val_table_ren_columns = mis_val_table_ren_columns[
        mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
    '% of Total Values', ascending=False).round(1)
    print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
        "There are " + str(mis_val_table_ren_columns.shape[0]) +
            " columns that have missing values.")
    return mis_val_table_ren_columns

missing_values_table(training_df)
```

```
Your selected dataframe has 50 columns.
There are 1 columns that have missing values.
```

Out[8]:

	Missing Values	% of Total Values
Age	9	2.3

Data Preparation

In [9]:

```
training_df['orderAgain'].value_counts()
```

Out[9]:

```
Yes    301
No      87
Name: orderAgain, dtype: int64
```

In [10]:

```
def impute_mode(X):
    return training_df[X].fillna(value=training_df[X].mode()[0], inplace=True)
```

```
impute_mode('Age')
```

In [11]:
```
training_df['Age'].isnull().any()
```

Out[11]:
```
False
```

In [41]:
```
import pandas_profiling
profile = training_df.profile_report(title="Pandas Profiling Report")
profile
```

Overview

Dataset statistics

Number of variables	50
Number of observations	388
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	64
Duplicate rows (%)	16.5%
Total size in memory	151.7 KiB
Average record size in memory	400.3 B

Variable types

Numeric	4
Categorical	45
Boolean	1

Alerts

Dataset has 64 (16.5%) duplicate rows	**Duplicates**
`Meal(P1)` is highly correlated with `Meal(P2)`	**High correlation**

Out[41]:

In [12]:
```
train_df, valid_df = train_test_split(training_df,test_size=0.3, stratify= training_df[
```

In [13]:
```
train_df.shape, valid_df.shape
```

Out[13]: ((271, 50), (117, 50))

In [14]:
```
train_df.reset_index(drop=True, inplace=True)
valid_df.reset_index(drop=True, inplace=True)
```

In [15]:
```
train_df.columns
```

Out[15]:
```
Index(['Age', 'Gender', 'Marital_Status', 'Occupation', 'Monthly_Income',
       'Educational_Qualifications', 'Family_size', 'latitude', 'longitude',
       'Meal(P1)', 'Meal(P2)', 'Perference(P1)', 'Perference(P2)',
       'Ease_and_convenient', 'Time_saving', 'More_restaurant_choices',
       'Easy_Payment_option', 'More_Offers_and_Discount', 'Good_Food_quality',
       'Good_Tracking_system', 'Self_Cooking', 'Health_Concern',
       'Late_Delivery', 'Poor_Hygiene', 'Bad_past_experience',
       'Unavailability', 'Unaffordable', 'Long_delivery_time',
       'Delay_of_delivery_person_getting_assigned',
       'Delay_of_delivery_person_picking_up_food', 'Wrong_order_delivered',
       'Missing_item', 'Order_placed_by_mistake', 'Influence_of_time',
       'Order_Time', 'Maximum_wait_time', 'Residence_in_busy_location',
       'Good_Road_Condition', 'Low_quantity_low_time',
       'Delivery_person_ability', 'Influence_of_rating', 'Less_Delivery_time',
       'High_Quality_of_package', 'Number_of_calls', 'Politeness', 'Freshness',
       'Temperature', 'Good_Taste', 'Good_Quantity', 'orderAgain'],
      dtype='object')
```

Spliting into X and y

In [16]:
```
X_train = train_df.drop(columns = ['orderAgain'],axis = 1)

y_train = train_df.orderAgain
y_train.replace({'Yes':1,'No':0},inplace = True)
```

In [17]:
```
X_valid = train_df.drop(columns = ['orderAgain'],axis = 1)
y_valid= train_df.orderAgain
y_valid.replace({'Yes':1,'No':0},inplace = True)
```

In [18]:
```
X_train.shape , y_train.shape , X_valid.shape , y_valid.shape
```

Out[18]:
```
((271, 49), (271,), (271, 49), (271,))
```

In [19]:
```
y_train.head()
```

Out[19]:
```
0    1
1    1
2    1
3    1
4    1
Name: orderAgain, dtype: int64
```

In [21]:
```
from sklearn.preprocessing import OrdinalEncoder

std = StandardScaler()
ord_enc = OrdinalEncoder()
preprocess = make_column_transformer((ord_enc, ['Gender', 'Marital_Status', 'Occupation
       'Educational_Qualifications', 'Family_size','Meal(P1)', 'Meal(P2)', 'Perference(
       'Ease_and_convenient', 'Time_saving', 'More_restaurant_choices',
       'Easy_Payment_option', 'More_Offers_and_Discount', 'Good_Food_quality',
       'Good_Tracking_system', 'Self_Cooking', 'Health_Concern',
       'Late_Delivery', 'Poor_Hygiene', 'Bad_past_experience',
       'Unavailability', 'Unaffordable', 'Long_delivery_time',
       'Delay_of_delivery_person_getting_assigned',
```

```
        'Delay_of_delivery_person_picking_up_food', 'Wrong_order_delivered',
        'Missing_item', 'Order_placed_by_mistake', 'Influence_of_time',
        'Order_Time', 'Maximum_wait_time', 'Residence_in_busy_location',
        'Good_Road_Condition', 'Low_quantity_low_time',
        'Delivery_person_ability', 'Influence_of_rating', 'Less_Delivery_time',
        'High_Quality_of_package', 'Number_of_calls', 'Politeness', 'Freshness',
        'Temperature', 'Good_Taste', 'Good_Quantity']),
(std, ['Age','latitude','longitude']), remainder = 'passthrough')
```

In [ ]:

Training Model

XG BOOST Classifier

In [31]:

```
xgb_clf = make_pipeline(
    preprocess,
    XGBClassifier())
```

In [38]:

```
xgb_clf.fit(X_train, y_train)

# Predicting on the test data
pred_test = xgb_clf.predict(X_valid)

#Calculating and printing the f1 score
f1_test = f1_score(y_valid, pred_test)
print('The f1 score for the testing data:', f1_test)

#Ploting the confusion matrix
conf_matrix(y_valid, pred_test)

lr_probs = xgb_clf.predict_proba(X_valid)
lr_probs = lr_probs[:, 1]

ns_probs = [0 for _ in range(len(y_valid))]

ns_auc = roc_auc_score(y_valid, ns_probs)
lr_auc = roc_auc_score(y_valid, lr_probs)

print('No Skill: ROC AUC=%.3f' % (ns_auc))
print('Logistic: ROC AUC=%.3f' % (lr_auc))

ns_fpr, ns_tpr, _ = roc_curve(y_valid, ns_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_valid, lr_probs)

# plot the roc curve for the model
plt.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
plt.plot(lr_fpr, lr_tpr, marker='.', label='XGBoost')

# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

# show the legend
plt.legend()
# show the plot
plt.show()
```

```
[14:33:37] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/lea
rner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the obj
ective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metr
ic if you'd like to restore the old behavior.

C:\Users\AKASH\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarning: The use
of label encoder in XGBClassifier is deprecated and will be removed in a future release.
To remove this warning, do the following: 1) Pass option use_label_encoder=False when co
nstructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with
0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
```

Out[38]:

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('ordinalencoder',
                                                  OrdinalEncoder(),
                                                  ['Gender', 'Marital_Status',
                                                   'Occupation',
                                                   'Monthly_Income',
                                                   'Educational_Qualifications',
                                                   'Family_size', 'Meal(P1)',
                                                   'Meal(P2)', 'Perference(P1)',
                                                   'Perference(P2)',
                                                   'Ease_and_convenient',
                                                   'Time_saving',
                                                   'More_restaurant_choices',
                                                   'Eas...
                               colsample_bytree=1, gamma=0, gpu_id=-1,
                               importance_type='gain',
                               interaction_constraints='',
                               learning_rate=0.300000012, max_delta_step=0,
                               max_depth=6, min_child_weight=1, missing=nan,
                               monotone_constraints='()', n_estimators=100,
                               n_jobs=12, num_parallel_tree=1, random_state=0,
                               reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                               subsample=1, tree_method='exact',
                               validate_parameters=1, verbosity=None))])
The f1 score for the testing data: 1.0
No Skill: ROC AUC=0.500
Logistic: ROC AUC=1.000
```

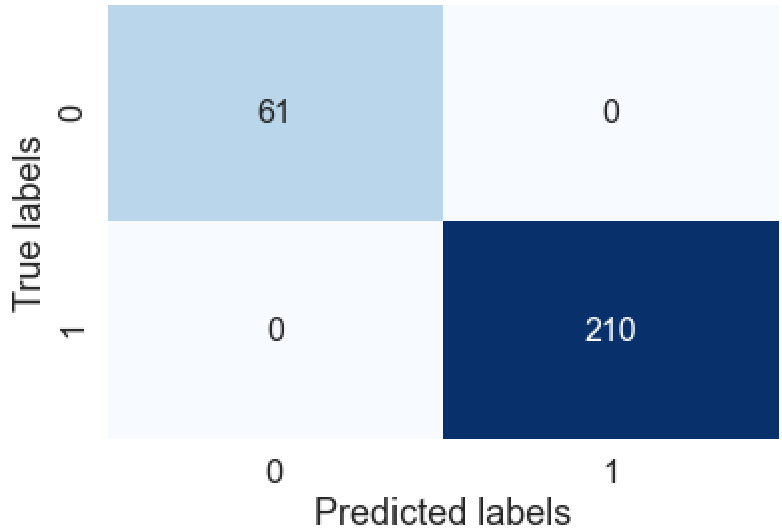
Out[38]: `[<matplotlib.lines.Line2D at 0x1df799cb400>]`

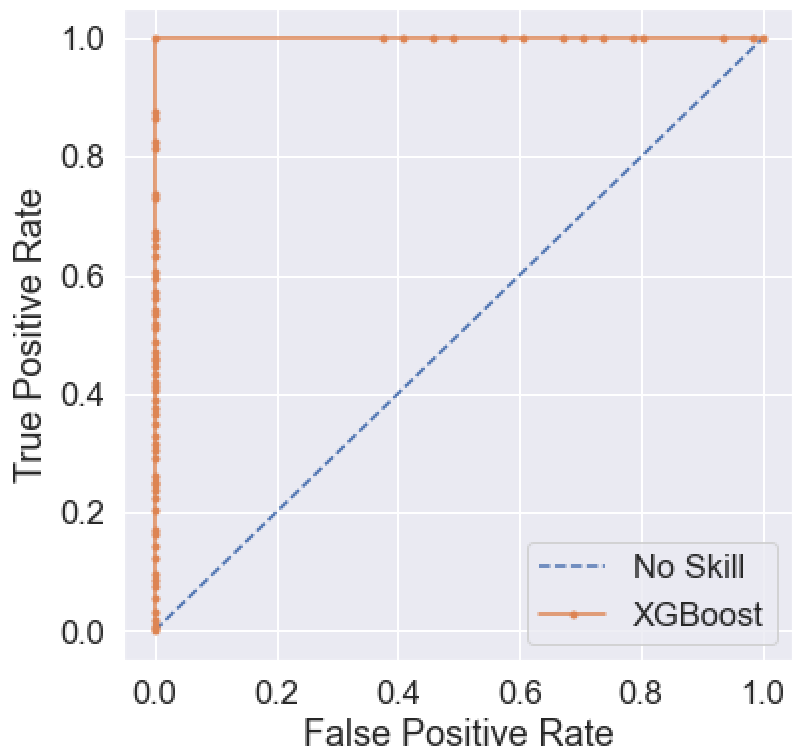
Out[38]: `[<matplotlib.lines.Line2D at 0x1df799cb880>]`

Out[38]: `Text(0.5, 0, 'False Positive Rate')`

Out[38]: `Text(0, 0.5, 'True Positive Rate')`

Out[38]: `<matplotlib.legend.Legend at 0x1df799cb7c0>`





Conclusion

Overall, the dataset had very little misisng value. We imputed the missing value with the mode.

The dataset was split on the strata of the target to ensure we had enough instances.

Ordinal Encoding was done to the variables to ensure it captures the weight in the repsonses. Strongly Agree being strongest and Strongly Disagree being weakest

Overall, we ran a simple classifier and a boosted classifier. Given the amount of information in the survey dataset the boosted model performs extremely well with good predicitve power.

In [ ]: