Data Exploration and Preprocess Prototyping

EDA notes

- No non-unique values in 'id' (no customer is represented more than once), drop 'id' column
- 310 NaN values in 'Arrival Delay in Minutes'
  - data dictionary does not explicitly address this, so I assume there was no delay in those instances
  - I will replace NaNs here with 0s
  - No other missing or null values found
- Intended target variable is very imbalanced
  Loyal Customer: 0.817322
  disloyal Customer: 0.182678
- Mean and standard deviation of survey answer features is fairly ubiquitous. Mean in particular hovers around 3 (scale of 1-5), indicating this particular subset of features is fairly balanced.
- 'Class' feature is represented by 3 categories, one of which is highly under-represented. It should also be noted that first-class flights are ***not*** represented at all. This should be kept in mind when interpreting the findings at the end of the project.
  Business: 0.477989 (dropped for OHE) Eco: 0.449886
  Eco Plus: 0.072124
- 'Travel type' is a little more than 2/3 business
- maximum delay on arrival *and* departure is ~26 hours, while minimum is exactly zero. . . as well as median and mode.

Load and inspect raw data

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

The data used is sourced from this kaggle dataset. It comes train-test split already, with 103,904 rows in the train set, and 26,000 rows in the test set, each row includes data on 23 candidate predictive features and one pre-determined target feature ('Customer Type').

In [2]:

```
df = pd.read_csv('../data/train.csv.zip',index_col=0 ,compression='zip')
df = df.drop('id',axis=True)
df.head()
```

Out[2]:

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking
0	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	3
1	Male	disloyal Customer	25	Business travel	Business	235	3	2	3
2	Female	Loyal Customer	26	Business travel	Business	1142	2	2	2
3	Female	Loyal Customer	25	Business travel	Business	562	2	5	5
4	Male	Loyal Customer	61	Business travel	Business	214	3	3	3

5 rows × 23 columns

In [3]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 103904 entries, 0 to 103903
Data columns (total 23 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   Gender                            103904 non-null  object 
 1   Customer Type                     103904 non-null  object 
 2   Age                               103904 non-null  int64  
 3   Type of Travel                    103904 non-null  object 
 4   Class                             103904 non-null  object 
 5   Flight Distance                   103904 non-null  int64  
 6   Inflight wifi service             103904 non-null  int64  
 7   Departure/Arrival time convenient  103904 non-null  int64  
 8   Ease of Online booking            103904 non-null  int64  
 9   Gate location                     103904 non-null  int64  
 10  Food and drink                    103904 non-null  int64  
 11  Online boarding                   103904 non-null  int64  
 12  Seat comfort                      103904 non-null  int64  
 13  Inflight entertainment            103904 non-null  int64  
 14  On-board service                  103904 non-null  int64  
 15  Leg room service                  103904 non-null  int64  
 16  Baggage handling                  103904 non-null  int64  
 17  Checkin service                   103904 non-null  int64  
 18  Inflight service                  103904 non-null  int64  
 19  Cleanliness                       103904 non-null  int64  
 20  Departure Delay in Minutes        103904 non-null  int64  
 21  Arrival Delay in Minutes          103594 non-null  float64
 22  satisfaction                      103904 non-null  object 
dtypes: float64(1), int64(17), object(5)
memory usage: 19.0+ MB
```

Handle missing values

310 NaN values for arrival delay feature. There are no NaN for depature delay, and both feauters contain many zeros (which is also the mode of both features). So I will be interpreting these NaN values as actually zero. I'm assuming that when there was no delay, there was no data input, so it would effectively be zero.

In [4]:
```
1 # 310 total NaN values in arrival delay column
2 df.loc[df['Arrival Delay in Minutes'].isna()].shape
3
4 # replace NaNs with zeros for arrival delay feature
5 df['Arrival Delay in Minutes'] = df['Arrival Delay in Minutes'].replace(
6
7 # check for any more nans
8 sum(df['Arrival Delay in Minutes'].isna())
```

Out[4]: 0

Descriptive Analysis and Exploration

Inspect object type features

Isolating features where the data type is 'object' and inspecting central tendencies and OHE efficacy.

In [5]:
```
# split objects and numerics
objx = ['Gender','Customer Type','Type of Travel','Class','satisfaction'
objx_df = df[objx]
objx_df.head()
```

Out[5]:

	Gender	Customer Type	Type of Travel	Class	satisfaction
0	Male	Loyal Customer	Personal Travel	Eco Plus	neutral or dissatisfied
1	Male	disloyal Customer	Business travel	Business	neutral or dissatisfied
2	Female	Loyal Customer	Business travel	Business	satisfied
3	Female	Loyal Customer	Business travel	Business	neutral or dissatisfied
4	Male	Loyal Customer	Business travel	Business	satisfied

In [6]:
```
objx_df.describe()
```

Out[6]:

	Gender	Customer Type	Type of Travel	Class	satisfaction
count	103904	103904	103904	103904	103904
unique	2	2	2	3	2
top	Female	Loyal Customer	Business travel	Business	neutral or dissatisfied
freq	52727	84923	71655	49665	58879

In [7]:
```
for col in objx_df.columns:
    print(df[col].value_counts(normalize=True))
    print("\n----------------------------------\n")
```

```
Female    0.507459
Male      0.492541
Name: Gender, dtype: float64

----------------------------------

Loyal Customer       0.817322
disloyal Customer    0.182678
Name: Customer Type, dtype: float64

----------------------------------

Business travel    0.689627
Personal Travel    0.310373
Name: Type of Travel, dtype: float64

----------------------------------

Business    0.477989
Eco         0.449886
Eco Plus    0.072124
Name: Class, dtype: float64

----------------------------------

neutral or dissatisfied    0.566667
satisfied                  0.433333
Name: satisfaction, dtype: float64

----------------------------------
```

observations on non-numeric data:

- Gender is very balanced, target (loyalty) is extremely imbalanced,
- Travel type is > 2/3 business,
- Ticket type is fairly balanced with one minority third class, first-class is not represented at all

Inspect Continuous Type Data

In [8]:
```
# split survey data and continuos data (flight metrics)
conts = ['Age','Flight Distance','Departure Delay in Minutes','Arrival De
cont_df = df[conts]
```

In [9]:
```
1 cont_df.head()
```

Out[9]:

	Age	Flight Distance	Departure Delay in Minutes	Arrival Delay in Minutes
0	13	460	25	18.0
1	25	235	1	6.0
2	26	1142	0	0.0
3	25	562	11	9.0
4	61	214	0	0.0

In [10]:
```
1 # good age range, good flight distance range, max delays < 30 min
2 cont_df.describe()
```

Out[10]:

	Age	Flight Distance	Departure Delay in Minutes	Arrival Delay in Minutes
count	103904.000000	103904.000000	103904.000000	103904.000000
mean	39.379706	1189.448375	14.815618	15.133392
std	15.114964	997.147281	38.230901	38.649776
min	7.000000	31.000000	0.000000	0.000000
25%	27.000000	414.000000	0.000000	0.000000
50%	40.000000	843.000000	0.000000	0.000000
75%	51.000000	1743.000000	12.000000	13.000000
max	85.000000	4983.000000	1592.000000	1584.000000

observations on continuous data: good age range, good flight distance range, max delays < 30 min

Inspect Survey Data (ordinal categorical)

In [11]:
```
ints_df = df.drop(objx,axis=1)
survey_df = df.drop('Customer Type',axis=1).iloc[:,5:19]
survey_df.head()
```

Out[11]:

	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment	C boa servi
0	3	4	3	1	5	3	5	5	
1	3	2	3	3	1	3	1	1	
2	2	2	2	2	5	5	5	5	
3	2	5	5	5	2	2	2	2	
4	3	3	3	3	4	5	5	3	

In [12]:
```
# looks good, looks normal
survey_df.describe()
```

Out[12]:

	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	bc
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.
mean	2.729683	3.060296	2.756901	2.976883	3.202129	3.
std	1.327829	1.525075	1.398929	1.277621	1.329533	1.
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.
25%	2.000000	2.000000	2.000000	2.000000	2.000000	2.
50%	3.000000	3.000000	3.000000	3.000000	3.000000	3.
75%	4.000000	4.000000	4.000000	4.000000	4.000000	4.
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.

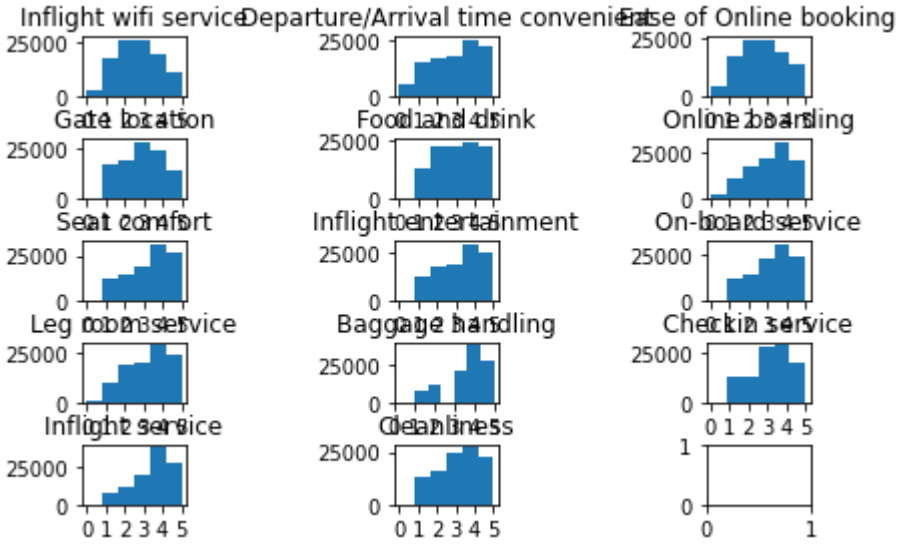
In [14]:

```
fig, axes = plt.subplots(5,3)
row = 0
column = 0

for col in survey_df.columns:
    axes[row,column].hist(survey_df[col],bins=6,align='mid')
    axes[row,column].set_title(col)
    axes[row,column].set_xticks([0,1,2,3,4,5])

    column += 1
    if column > 2:
        row += 1
        column = 0

# plt.set_size_inches((15,8))
plt.tight_layout()
plt.subplots_adjust(hspace=0.7,wspace=2.0)
plt.show()
```



In [15]:
```
# lots of N/A answers
np.sum(survey_df==0)
```

Out[15]:
```
Inflight wifi service                3103
Departure/Arrival time convenient    5300
Ease of Online booking               4487
Gate location                           1
Food and drink                        107
Online boarding                      2428
Seat comfort                            1
Inflight entertainment                 14
On-board service                        3
Leg room service                      472
Baggage handling                        0
Checkin service                         1
Inflight service                        3
Cleanliness                            12
dtype: int64
```

In [16]:
```
# number of rows containing a zero
zeros = 0
for row in survey_df.iterrows():
    zero_bool = (row[1]==0).sum()
    if zero_bool > 0:
        zeros += 1
print("Rows with zeros in survey:", zeros)
```

```
Rows with zeros in survey: 8200
```

In [17]:
```
# this loop replaces any zeros with the mode of the row
for row in survey_df[0:25].iterrows():
    zero_bool = (row[1]==0).sum()
    row_mode = row[1].aggregate(func='mode')
    if zero_bool > 0:
        print(row)
        print(survey_df.iloc[row[0]].replace(0,row_mode[0]))
```

```
(24, Inflight wifi service                 5
Departure/Arrival time convenient    0
Ease of Online booking               5
Gate location                        1
Food and drink                       1
Online boarding                      5
Seat comfort                         1
Inflight entertainment               1
On-board service                     4
Leg room service                     5
Baggage handling                     5
Checkin service                      3
Inflight service                     5
Cleanliness                          1
Name: 24, dtype: int64)
Inflight wifi service                5
Departure/Arrival time convenient    5
Ease of Online booking               5
Gate location                        1
Food and drink                       1
Online boarding                      5
Seat comfort                         1
Inflight entertainment               1
On-board service                     4
Leg room service                     5
Baggage handling                     5
Checkin service                      3
Inflight service                     5
Cleanliness                          1
Name: 24, dtype: int64
```

In [18]:
```
bool_test = survey_df
zeros = 0
for row in bool_test[0:50].iterrows():
    zero_bool = (row[1]==0).sum()
    if zero_bool > 0:
        zeros += 1
print("Rows with zeros in survey:", zeros)
```

```
Rows with zeros in survey: 2
```

In [19]:
```
for row in survey_df[0:50].iterrows():
    zero_bool = (row[1]==0).sum()
    row_mode = row[1].aggregate(func='mode')

    if zero_bool > 0:
        survey_df.iloc[row[0]].replace(0,row_mode[0],inplace=True) # if t

    for col in survey_df.columns:
        bool_test[col]=survey_df[col] # replace columns in X with respect
```

In [21]:
```
np.sum(bool_test[0:50]==0)
```

Out[21]:
```
Inflight wifi service                0
Departure/Arrival time convenient    0
Ease of Online booking               0
Gate location                        0
Food and drink                       0
Online boarding                      0
Seat comfort                         0
Inflight entertainment               0
On-board service                     0
Leg room service                     0
Baggage handling                     0
Checkin service                      0
Inflight service                     0
Cleanliness                          0
dtype: int64
```

observations on sruvey data:

- survey data appears relatively normally distributed. Some transformation may help.
- central tendencies here are fairly consistent accross the board.
- 0 represent "N/A" answers. consider dropping as outliers or imputing.

Preprocess Train-Test Split, Build Baseline Model (logistic regression)

In [17]:
```
objx = ['Gender','Type of Travel','Class','satisfaction']

train_df = pd.read_csv('../data/train.csv.zip',compression='zip',index_c
train_df.drop('id',axis=1,inplace=True)

test_df = pd.read_csv('../data/test.csv.zip',compression='zip',index_col
test_df.drop('id',axis=1,inplace=True)
```

In [18]:
```
# clean and OHE training set
X_train = train_df.drop('Customer Type',axis=1)
X_train['Arrival Delay in Minutes'] = X_train['Arrival Delay in Minutes'
X_train = pd.get_dummies(X_train,drop_first=True) # one hot encoding obje

y_train = train_df['Customer Type']
y_train = pd.get_dummies(y_train,drop_first=True)['disloyal Customer'] #
```

In [19]:
```
# impute zeros for survey data
survey_labels = list(survey_df.columns)
survey_train = X_train[survey_labels]

for row in survey_train.iterrows():
    zero_bool = (row[1]==0).sum()
    row_mode = row[1].aggregate(func='mode')

    if zero_bool > 0:
        survey_train.iloc[row[0]].replace(0,row_mode[0],inplace=True)

for col in survey_labels:
    X_train[col] = survey_train[col]
```

In [20]:
```
# clean and OHE test set
X_test = test_df.drop('Customer Type',axis=1)
X_test['Arrival Delay in Minutes'] = X_test['Arrival Delay in Minutes'].
X_test = pd.get_dummies(X_test,drop_first=True) # one hot encoding objec

y_test = test_df['Customer Type']
y_test = pd.get_dummies(y_test,drop_first=True)['disloyal Customer'] # 1
```

In [21]:
```
# impute zeros for survey data
survey_test = X_test[survey_labels]

for row in survey_test.iterrows():
    zero_bool = (row[1]==0).sum()
    row_mode = row[1].aggregate(func='mode')

    if zero_bool > 0:
        survey_test.iloc[row[0]].replace(0,row_mode[0],inplace=True)

for col in survey_labels:
    X_test[col]=survey_test[col]
```

In [22]:
```
# verifying imputation succeded
X_train.iloc[24]
```

Out[22]:
```
Age                                    23.0
Flight Distance                       452.0
Inflight wifi service                   5.0
Departure/Arrival time convenient       5.0
Ease of Online booking                  5.0
Gate location                           1.0
Food and drink                          1.0
Online boarding                         5.0
Seat comfort                            1.0
Inflight entertainment                  1.0
On-board service                        4.0
Leg room service                        5.0
Baggage handling                        5.0
Checkin service                         3.0
Inflight service                        5.0
Cleanliness                             1.0
Departure Delay in Minutes             54.0
Arrival Delay in Minutes               44.0
Gender_Male                             0.0
Type of Travel_Personal Travel          0.0
Class_Eco                               1.0
Class_Eco Plus                          0.0
satisfaction_satisfied                  1.0
Name: 24, dtype: float64
```

train baseline model(s)

In [23]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import plot_confusion_matrix, classification_report
```

In [24]:
```
logreg = LogisticRegression(solver='liblinear')
logreg.fit(X_train,y_train)

# generate y hat
train_pred = logreg.predict(X_train)
test_pred = logreg.predict(X_test)
```

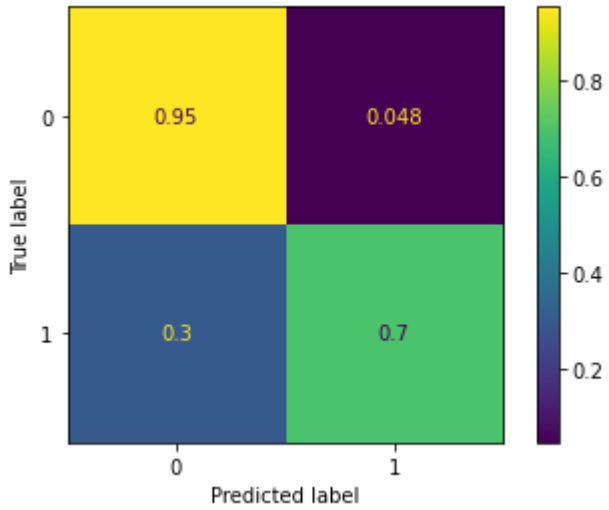
In [25]:

```
# train set prediction performance
plot_confusion_matrix(logreg,X_train,y_train,normalize='true')
print(classification_report(y_train,train_pred))
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     84923
           1       0.76      0.70      0.73     18981

    accuracy                           0.91    103904
   macro avg       0.85      0.82      0.84    103904
weighted avg       0.90      0.91      0.90    103904
```

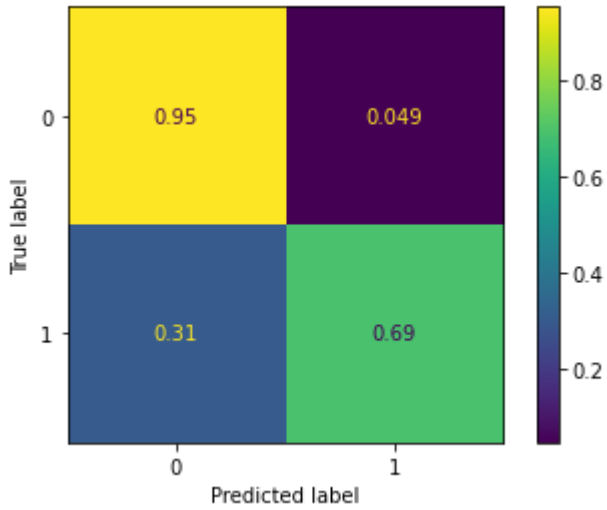


In [26]:
```
# test set prediction performance
plot_confusion_matrix(logreg,X_test,y_test,normalize='true')
print(classification_report(y_test,test_pred))
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     21177
           1       0.76      0.69      0.73      4799

    accuracy                           0.90     25976
   macro avg       0.85      0.82      0.83     25976
weighted avg       0.90      0.90      0.90     25976
```



In [27]:
```
from sklearn.model_selection import cross_val_score

cv_train_score = cross_val_score(logreg,X_train,y_train,cv=5,scoring='f1
cv_train_score.mean()
```

Out[27]: 0.7270997263421876

In [28]:
```
cv_test_score = cross_val_score(logreg,X_test,y_test,cv=5,scoring='f1')
cv_test_score.mean()
```

Out[28]: 0.7226140314201694

In [29]:
```
abs(cv_train_score.mean() - cv_test_score.mean())
```

Out[29]: 0.004485694922018113

Final EDA considerations: Baseline model missclassifies the target class about 30% of the time. A great starting point considering the class imbalance.

In the next notebook I will explore more advanced preprocessing techniques and hyperparameter some tuning.

My broader plan is to use logistic regression to discover the ideal preprocessing to handle the class imbalance, and then build a stronger decision tree and then random forest. The random forest should be the model that actually gets deployed.