

Flatiron School DS-Flex Phase Three Milestone

Business Case Overview

The Company

Sigma Airlines is the leading consumer aviation transportation service, providing the best value in customer experience and comfort. In order to continue leading the industry in customer satisfaction Sigma Airlines is taking steps to make more data-driven decisions around how to continually improve the way its customers are served.

The Use Case

The Sigma Airlines E.V.P. and Chief Customer Service Officer, Madison Housend, wishes to better understand the differences in behaviour and impression of services rendered, between habitual return ('loyal') and non-return ('disloyal') customers. Housend wishes to understand what services and/or aspects of the customer experience need improvement the most in order to retain more customer loyalty (i.e. increase churn rate of disloyal-to-loyal customers).

Data Overview

The data used for the ML development involved in this analysis is sourced from this [kaggle](#) dataset. It is a dataset of over 100k records. Each record represents an individual customer, the details of their flight (e.g. distance, delay time), and survey responses regarding their satisfaction with the services rendered while boarding, in-flight and deboarding.

The details of the preprocessing steps taken for this project are provided in the [EDA notebook](#) ([./development/EDA.ipynb](#)) and [resampling notebook](#) ([./development/resampling.ipynb](#)), which are both located in the development/ folder of this repository. Additionally detailed descriptions of the *raw data* features and their summary statistics (ranges, averages, etc.) can be found in the [data dictionary](#) ([./data/dictionary.md](#)) located in the data/ folder.

To summarize:

- The selected target feature is the column titled "Customer Type".
- The target feature is binary (Loyal/disloyal), and the target class is "disloyal". Disloyal is represented as 1, after one-hot-encoding.
- The target class is highly imbalanced with only about 18% of the raw training data being labeled as disloyal. The training data only is resampled using SMOTEENN, and ensemble of oversampling the target class with SMOTE and undersampling the non-target class with ENN.
- There are 4 features used that contain continuous data, providing details of the flight itself.

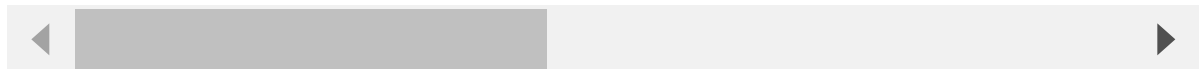
- There are another 18 features used that contain categorical data providing information on the customer themselves and their trip (e.g. gender, purpose for travel) as well as the survey responses the customer provided.

```
In [1]: 1 import pandas as pd
        2 raw_df = pd.read_csv("../data/train.csv.zip", compression='zip', index_col=0)
        3 raw_df.head()
        4
```

Out[1]:

	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



Data Understanding

Data Cleaning and Preprocessing

The code used to execute the preprocessing has been packaged into a module called [preprocessor \(./development/preprocessor.py\)](#), located in the development/ folder along with other ML and EDA development notebooks. The preprocessor module contains two functions, `data_cleaner()` and `data_sampler()`.

The `data_cleaner()` function is commented step by step in the preprocessor module linked above. The process of developing this function can be found in the [EDA notebook](#) also linked above. The `data_cleaner()` highlights are:

- The survey is on a scale of 1-5, some zeros were present which were being interpreted as NaN values. The data description found on Kaggle with the data source states that these were "N/A" answers. I imputed these zeros as the mode of the rest of the survey answers in a given row, based on the assumption that whatever a surveyee's most common answer is, it is more likely they would provide that same answer than any other value for a given survey question. This technique *does not eliminate all the zeros* but it reduces the total records within the survey features that contain zeros from over eight thousand to less than three hundred. My intentions here were to deal with the NaN issue, while maintaining the integrity of the scaling

of the survey data (1-5), as well as keeping the imputed survey responses aligned with the disposition of the relevant surveyee, I suspect that the rows that still have zeros after data cleaning are such because most of the answers provide for that record were in fact zero.

- Some arrival delays were NaN values. Considering that departure delays had zero NaNs but many zeros. I made the assumption that these NaN values were in fact flights were there was not a delay (so no delay time was recorded). So to eliminate NaNs, again for the sake of modeling, while keeping consistent with the nature of the raw data I replaced the missing values here simply with numerical zeros.
- Lastly, I performed one-hot-encoding (using pandas) on categorical features including the target feature, *but not the survey features*. I chose to not OHE the survey because although it is categorical it is numerical and *ordinal* meaning there is a significance of scale between one answer choice and the others, not simply a boolean relationship. (i.e. 4 is greater than 3 and even more so than 2, not simply just not 3 and just not 2).

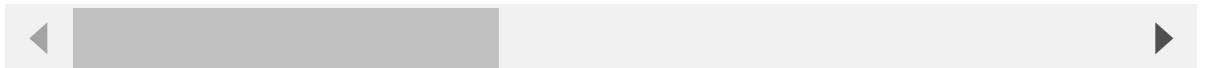
```
In [2]: 1 import seaborn as sns
        2 import numpy as np
        3 import matplotlib.pyplot as plt
        4 import development.preprocessor as pre
        5
        6 X_train, y_train = pre.data_cleaner("./data/train.csv.zip")
        7 X_test, y_test = pre.data_cleaner("./data/test.csv.zip")
        8
```

```
In [3]: 1 # raw training data preview
        2 X_train.head()
```

Out[3]:

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	entertainment
0	13	460	3	4	3	1	5	3	5	
1	25	235	3	2	3	3	1	3	1	
2	26	1142	2	2	2	2	5	5	5	
3	25	562	2	5	5	5	2	2	2	
4	61	214	3	3	3	3	4	5	5	

5 rows × 23 columns



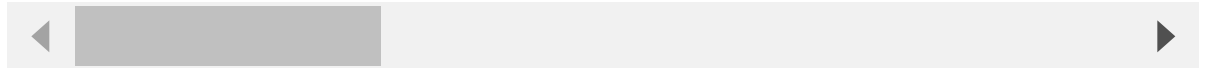
In [4]:

```
1 # Summary statistics on training data
2 X_train.describe()
```

Out[4]:

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000
mean	39.379706	1189.448375	2.816985	3.213707	2.884239	2.816985
std	15.114964	997.147281	1.267507	1.390750	1.307256	1.267507
min	7.000000	31.000000	0.000000	0.000000	0.000000	1.000000
25%	27.000000	414.000000	2.000000	2.000000	2.000000	2.000000
50%	40.000000	843.000000	3.000000	3.000000	3.000000	3.000000
75%	51.000000	1743.000000	4.000000	4.000000	4.000000	4.000000
max	85.000000	4983.000000	5.000000	5.000000	5.000000	5.000000

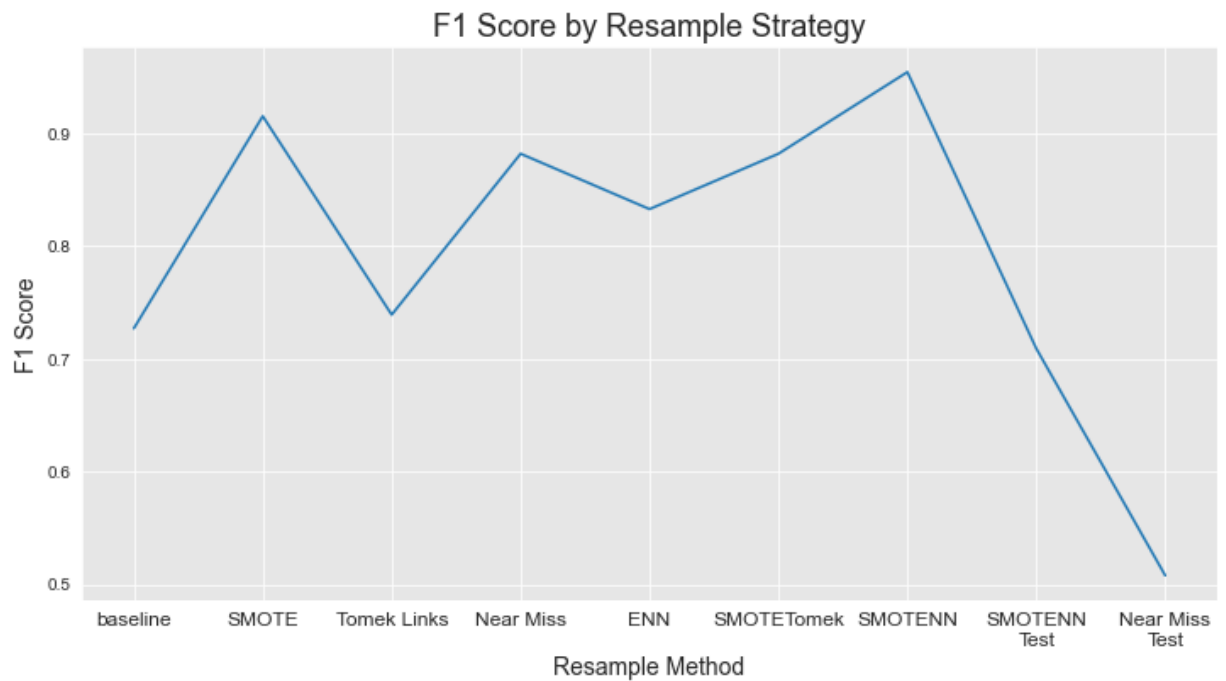
8 rows × 7 columns



Data Resampling

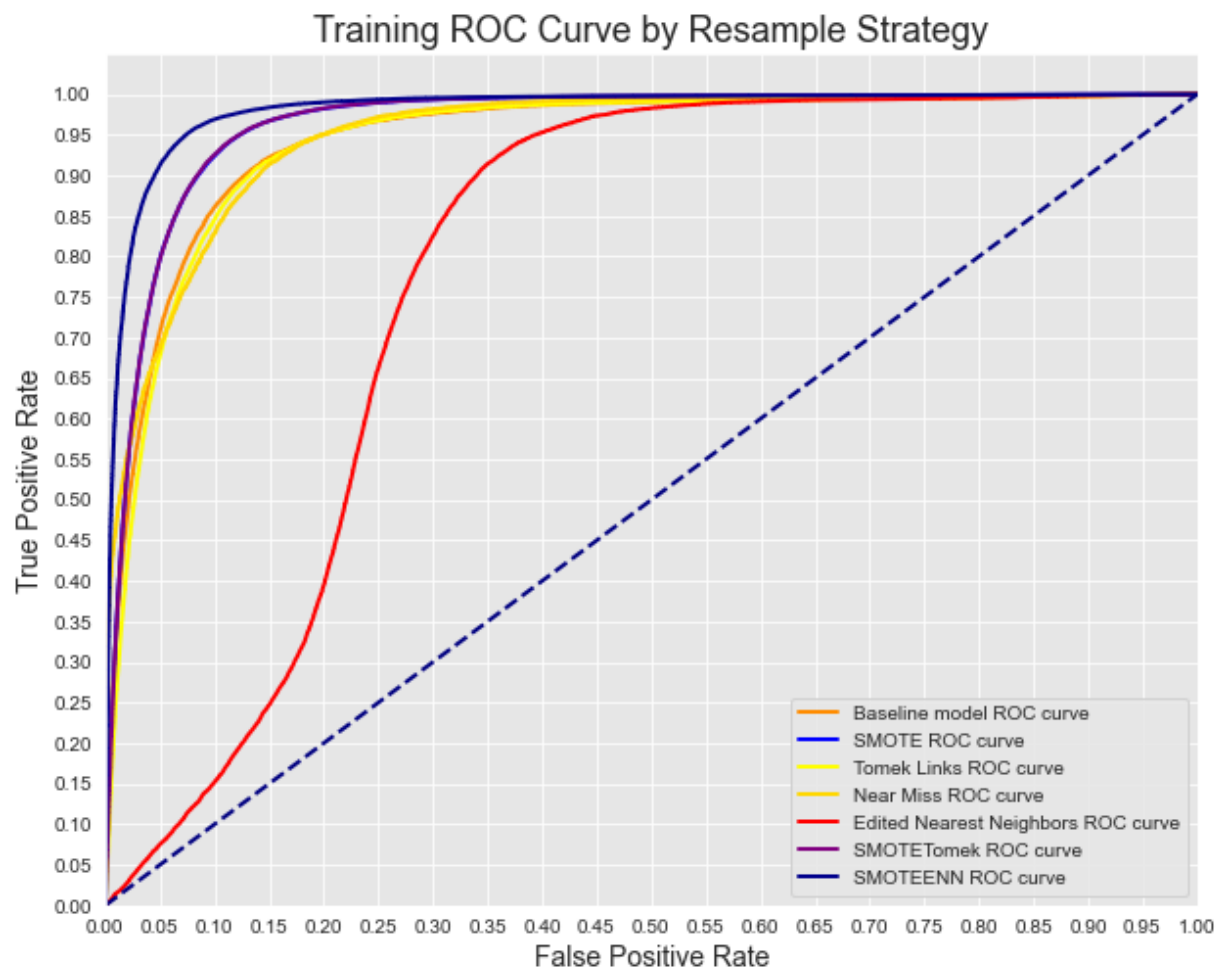
In the resampling notebook (linked in the overview section) I trained a series of logistic regression models. The only difference between the models is how the method of resampling used on the training data. The hyperparameters were all the same, the test data was not resampled but was cleaned using the `data_cleaner()` function.

I selected the SMOTEENN algorithm based mainly on its impact on the model's F1 score, shown here:

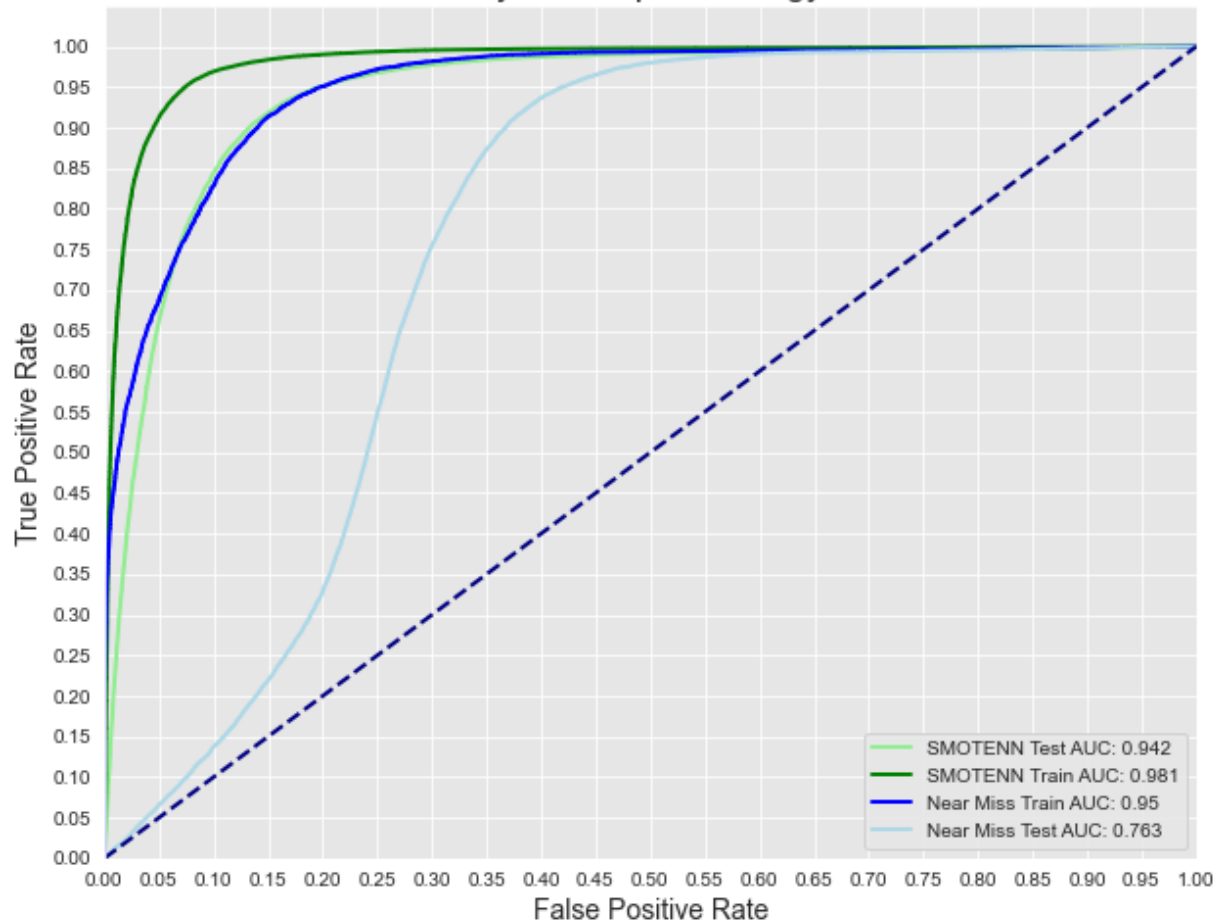


As well as its impact on the area under the model's receiver operating characteristic curve (ROC AUC). The ROC AUC on the models' fits of the training data showed SMOTEENN to be the highest performer:

Fig. 1|Fig. 2 -|-



ROC Curve Train vs Test by Resample Strategy



After evaluating the highest performing ensemble method (SMOTEENN) and the highest performing non-ensemble method (Near Miss) against the test data set, there was a significant decrease in F1-score by both models (near miss more so). Upon inspection of the testing ROC AUC of each candidate it was obvious that SMOTEENN was able to generalize to unseen data much better than Near Miss. Simply put, the SMOTEENN model performed on the test data as well as the Near Miss model performed only on the training data (see fig. 2). With these discoveries I selected SMOTEENN as my resampling method for the remainder of the project and wrote the `data_sampler()` function to make the resampling method portable and consistent between notebooks.

The `data_sampler()` function has minimal commenting, however the logic flow is extremely simple and the `imblearn` library documentation can provide robust explanations of the algorithms being utilized. After an exploratory analysis of various resampling methods passed into a very basic logistic regression model I selected the SMOTEENN algorithm based on its affect on F1-score and ROC AUC compared to other resampling algorithms. I only wrote a function to perform this step so that I could manually set the sampling strategy used once.

The [SMOTEENN \(https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTEENN.html\)](https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTEENN.html) algorithm ensembles oversampling with [Synthetic Minority OvEr-sampling \(https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html?highlight=smote#imblearn.over_sampling.SMOTE\)](https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html?highlight=smote#imblearn.over_sampling.SMOTE) which uses a distance based algorithm to synthesize new data points that are "close" to the original data points, and undersampling the

majority class with [Edited Nearest Neighbors \(https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling EditedNearestNeighbours.html?highlight=enn\)](https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling EditedNearestNeighbours.html?highlight=enn) which uses a distance based algorithm that takes the k-nearest data points to a minority data point and eliminates any of those neighboring observations that belong to the majority class.

Its important for me to emphasize that **only the training data is resampled** I do not resample the testing data, I only clean it using the `data_cleaner()` function. So the train and test data are cleaned the exact same way, but the training data retains its original class imbalance (which is relatively proportional to imbalance of the raw training data)

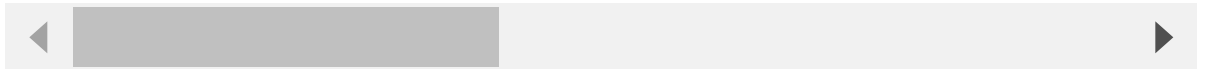
```
In [5]: 1 X_resample, y_resample = pre.data_sampler(X_train,y_train)
```

```
In [6]: 1 X_resample.head()
```

Out[6]:

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	entertainment
0	13	460	3	4	3	1	5	3	5	
1	61	214	3	3	3	3	4	5	5	
2	47	1276	2	4	2	3	2	2	2	
3	52	2035	4	3	4	4	5	5	5	
4	12	308	2	4	2	2	1	2	1	

5 rows × 23 columns



In [7]:

▶

1 X_resample.describe()

Out[7]:

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location
count	137185.000000	137185.000000	137185.000000	137185.000000	137185.000000	137185.000000
mean	35.880716	1050.610103	2.652498	2.868229	2.697365	2.697365
std	14.409724	914.827333	1.201290	1.293327	1.229297	1.229297
min	7.000000	31.000000	0.000000	0.000000	0.000000	0.000000
25%	24.000000	393.000000	2.000000	2.000000	2.000000	2.000000
50%	35.000000	736.000000	3.000000	3.000000	3.000000	3.000000
75%	46.000000	1389.000000	4.000000	4.000000	4.000000	4.000000
max	85.000000	4983.000000	5.000000	5.000000	5.000000	5.000000

8 rows × 23 columns

◀

▶

Visualizations of the Impact of Resampling with SMOTEENN

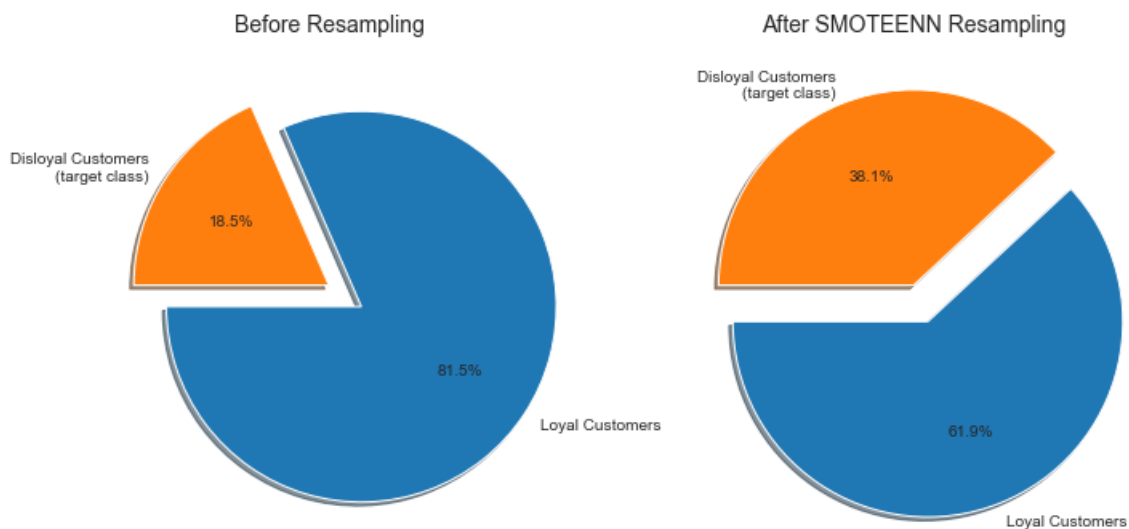
These data visualization below illustrate the impact of the SMOTEENN resampling algorithm on the relative weights and distributions of values and/or classes amongst the various features.


```

In [8]: 1 sns.set_style('darkgrid', {'axes.facecolor': '0.9'})
2 fig,axs = plt.subplots(1,2)
3 axs[0].pie(y_test.value_counts(normalize=True),
4           labels=['Loyal Customers','Disloyal Customers\n(target class)'],
5           startangle=180,
6           shadow=True,
7           explode=[.2,0],
8           autopct='%1.1f%%')
9 axs[0].set_title("Before Resampling",fontsize=14)
10
11 axs[1].pie(y_resample.value_counts(normalize=True),
12           labels=['Loyal Customers','Disloyal Customers\n(target class)'],
13           startangle=180,
14           shadow=True,
15           explode=[.2,0],
16           autopct='%1.1f%%')
17 axs[1].set_title("After SMOTEENN Resampling",fontsize=14)
18
19 fig.suptitle("\n\n\n\n\n\nTarget Feature Class Distribution",fontsize=18)
20 fig.set_size_inches(12,11)
21 plt.subplots_adjust(wspace=0.2)
22 plt.show()

```

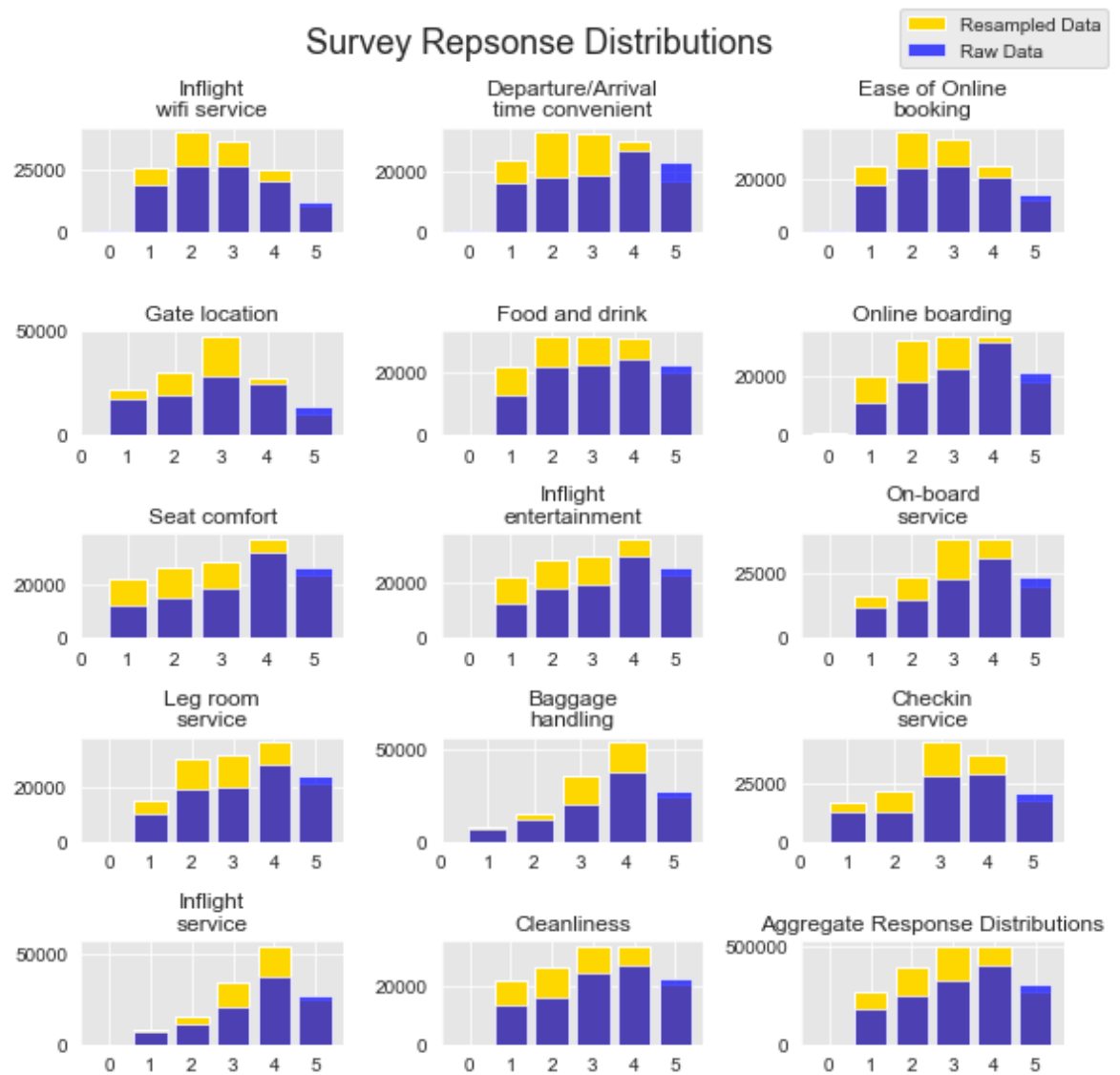
Target Feature Class Distribution



```

In [9]: 1 survey_df = X_train.iloc[:,2:16]
2 resample_survey_df = X_resample.iloc[:,2:16]
3
4 survey_labels = [
5     'Inflight\nwifi service',
6     'Departure/Arrival\ntime convenient',
7     'Ease of Online\nbooking',
8     'Gate location',
9     'Food and drink',
10    'Online boarding',
11    'Seat comfort',
12    'Inflight\nentertainment',
13    'On-board\nservice',
14    'Leg room\nservice',
15    'Baggage\nhandling',
16    'Checkin\nservice',
17    'Inflight\nservice',
18    'Cleanliness']
19
20 fig, axes = plt.subplots(5,3)
21 row = 0
22 column = 0
23
24 for idx,col in enumerate(resample_survey_df.columns):
25
26     raw = survey_df[col].value_counts().to_dict()
27     resample = resample_survey_df[col].value_counts().to_dict()
28
29     axes[row,column].bar(x=list(resample.keys()),height=list(resample.values()))
30     axes[row,column].bar(x=list(raw.keys()),height=list(raw.values()),color='red')
31     axes[row,column].set_title(survey_labels[idx])
32     axes[row,column].set_xticks([0,1,2,3,4,5])
33
34     column += 1
35     if column > 2:
36         row += 1
37         column = 0
38
39 survey_count_dict = {1:0,2:0,3:0,4:0,5:0,0:0}
40 resample_survey_count_dict = {1:0,2:0,3:0,4:0,5:0,0:0}
41
42 for key in survey_count_dict.keys():
43     survey_count_dict[key] += (survey_df.values==key).sum()
44     resample_survey_count_dict[key] += (resample_survey_df.values==key).sum()
45
46 axes[4,2].bar(x=list(resample_survey_count_dict.keys()),height=list(resample_survey_count_dict.values()))
47 axes[4,2].bar(x=list(survey_count_dict.keys()),height=list(survey_count_dict.values()),color='red')
48 axes[4,2].set_title("Aggregate Response Distributions")
49 axes[4,2].set_xticks([0,1,2,3,4,5])
50
51 fig.set_size_inches(8, 8)
52 fig.suptitle("Survey Response Distributions",fontsize=18)
53 fig.tight_layout()
54 fig.legend(loc='upper right',labels=['Resampled Data','Raw Data'])
55 plt.show()

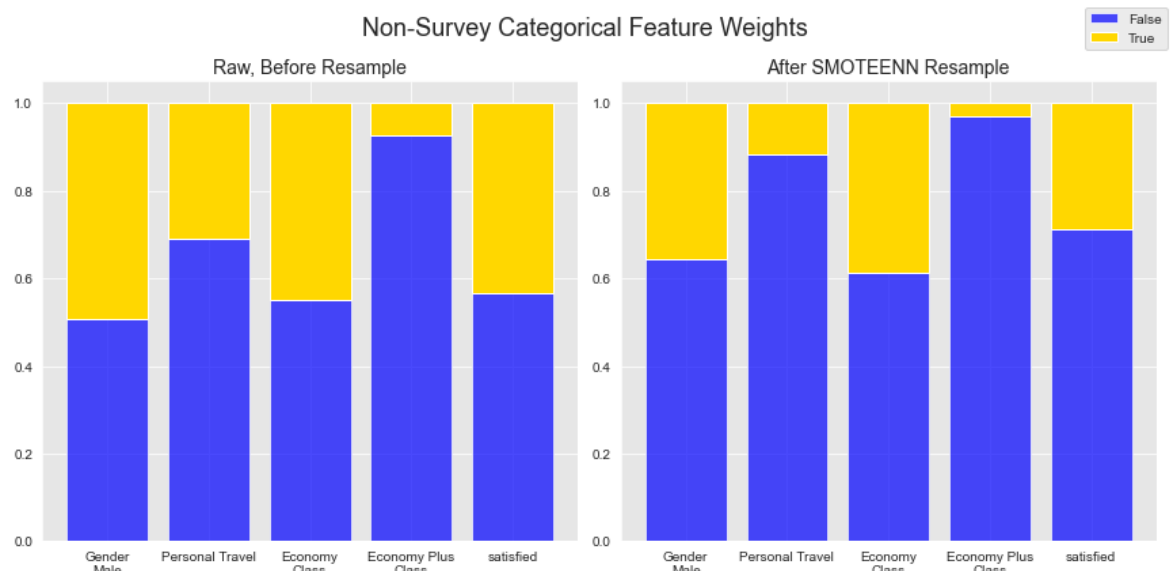
```



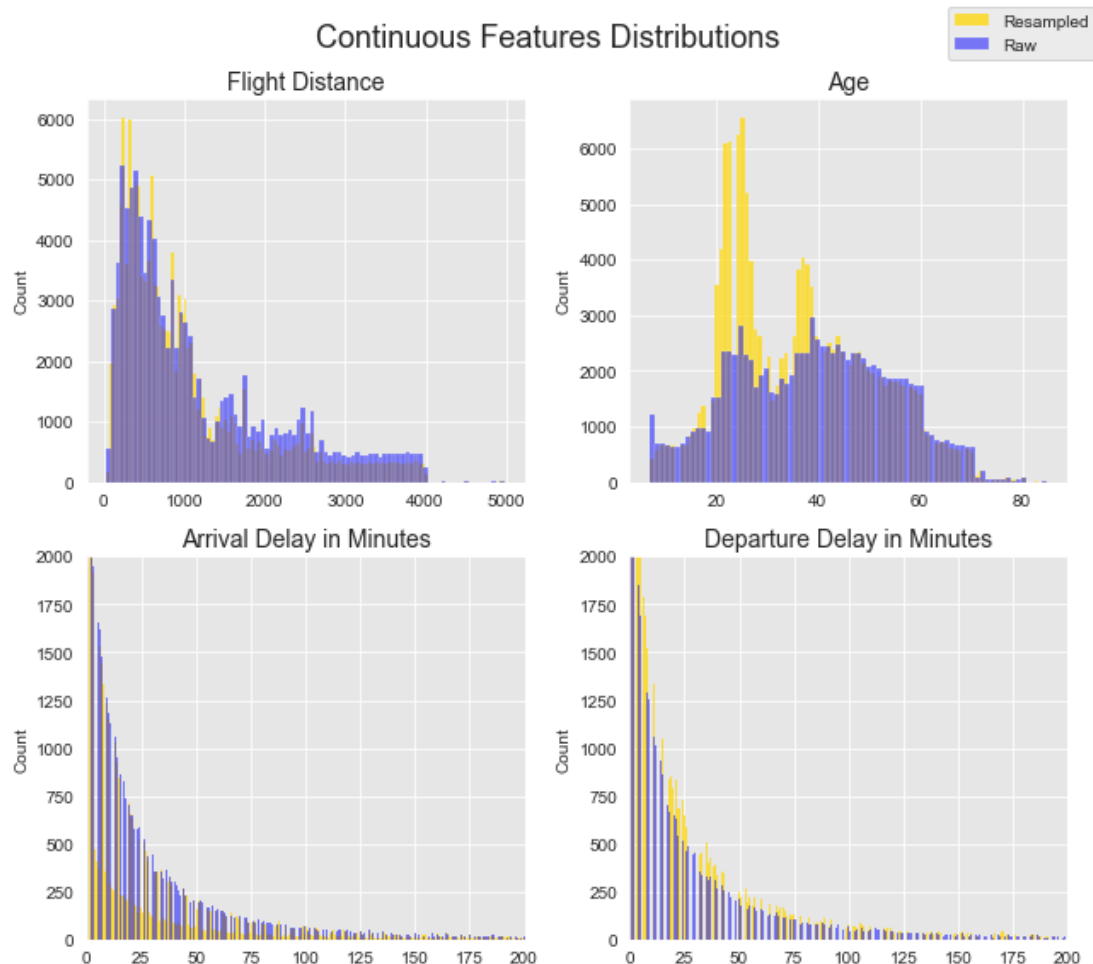
```

In [10]: 1 features = ['Gender_Male', 'Type of Travel_Personal Travel', 'Class_Eco', 'Class_Eco Plus', 'satisfied']
2 labels = ['Gender\nMale', 'Personal Travel', 'Economy\nClass', 'Economy Plus', 'satisfied']
3
4 raw_ones = []
5 raw_zeros = []
6
7 for idx,label in enumerate(labels):
8     series = X_train[features[idx]].value_counts(normalize=True)
9     raw_ones.append(series[1])
10    raw_zeros.append(series[0])
11
12 resample_ones = []
13 resample_zeros = []
14
15 for idx,label in enumerate(labels):
16     series = X_resample[features[idx]].value_counts(normalize=True)
17     resample_ones.append(series[1])
18     resample_zeros.append(series[0])
19
20 # fig.set_size_inches(8, 8)
21 fig,axs = plt.subplots(1,2)
22 axs[0].bar(labels,raw_zeros,label='False',color='blue',alpha=.7)
23 axs[0].bar(labels,raw_ones,label='True',bottom=raw_zeros,color='gold')
24 axs[0].set_title('Raw, Before Resample',fontsize=14)
25
26 axs[1].bar(labels,resample_zeros,label='False',color='blue',alpha=.7)
27 axs[1].bar(labels,resample_ones,label='True',bottom=resample_zeros,color='gold')
28 axs[1].set_title('After SMOTEENN Resample',fontsize=14)
29
30 fig.legend(['False', 'True'])
31 fig.set_size_inches(12, 6)
32 fig.suptitle("Non-Survey Categorical Feature Weights",fontsize=18)
33 fig.tight_layout()
34 plt.show()

```




```
In [11]: 1 fig,axs = plt.subplots(2,2)
2
3 #Flight distance
4 ff = sns.histplot(data=X_resample['Flight Distance'],ax=axs[0,0],color='gold')
5 f = sns.histplot(data=X_train['Flight Distance'],ax=axs[0,0],color='blue',alpha=.5)
6 axs[0,0].set_xlabel("")
7
8 #age
9 aa = sns.histplot(data=X_resample['Age'],ax=axs[0,1],color='gold')
10 a = sns.histplot(data=X_train['Age'],ax=axs[0,1],color='blue',alpha=.5)
11 axs[0,1].set_xlabel("")
12
13 #Arrival Delay in Minutes
14 dede = sns.histplot(data=X_resample['Arrival Delay in Minutes'],ax=axs[1,0],color='gold')
15 de = sns.histplot(data=X_train['Arrival Delay in Minutes'],ax=axs[1,0],color='blue',alpha=.5)
16 axs[1,0].set_xlabel("")
17 axs[1,0].set_ylim(0,2000)
18 axs[1,0].set_xlim(0,200)
19
20 #Departure Delay in Minutes
21 dldl = sns.histplot(data=X_resample['Departure Delay in Minutes'],ax=axs[1,1],color='gold')
22 dl = sns.histplot(data=X_train['Departure Delay in Minutes'],ax=axs[1,1],color='blue',alpha=.5)
23 axs[1,1].set_xlabel("")
24 axs[1,1].set_ylim(0,2000)
25 axs[1,1].set_xlim(0,200)
26
27 fig.suptitle("Continuous Features Distributions",fontsize=18)
28 fig.legend(['Resampled','Raw'])
29 fig.set_size_inches(9, 8)
30 fig.tight_layout()
31 plt.show()
```

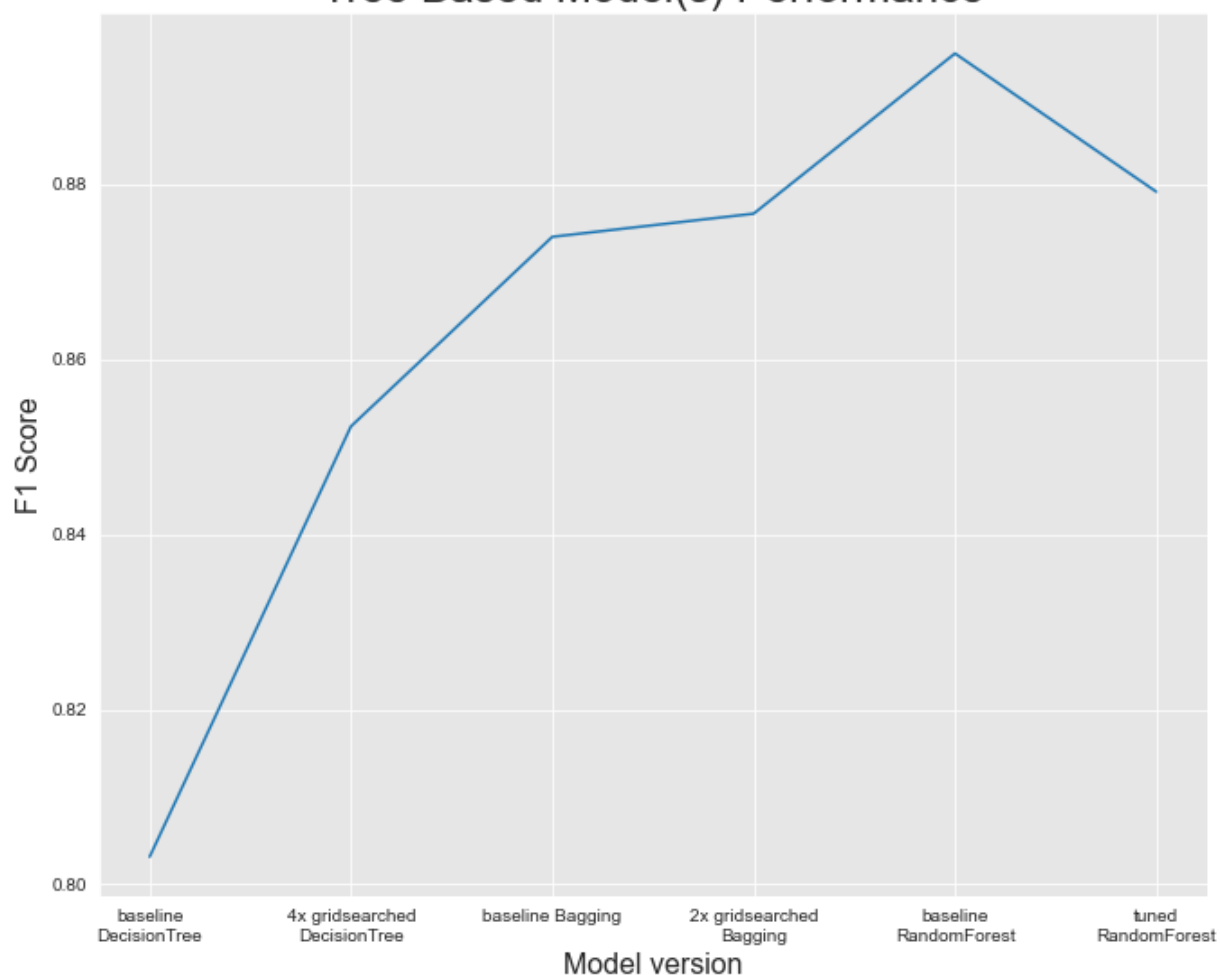


Tree-Based Model Development and Selection

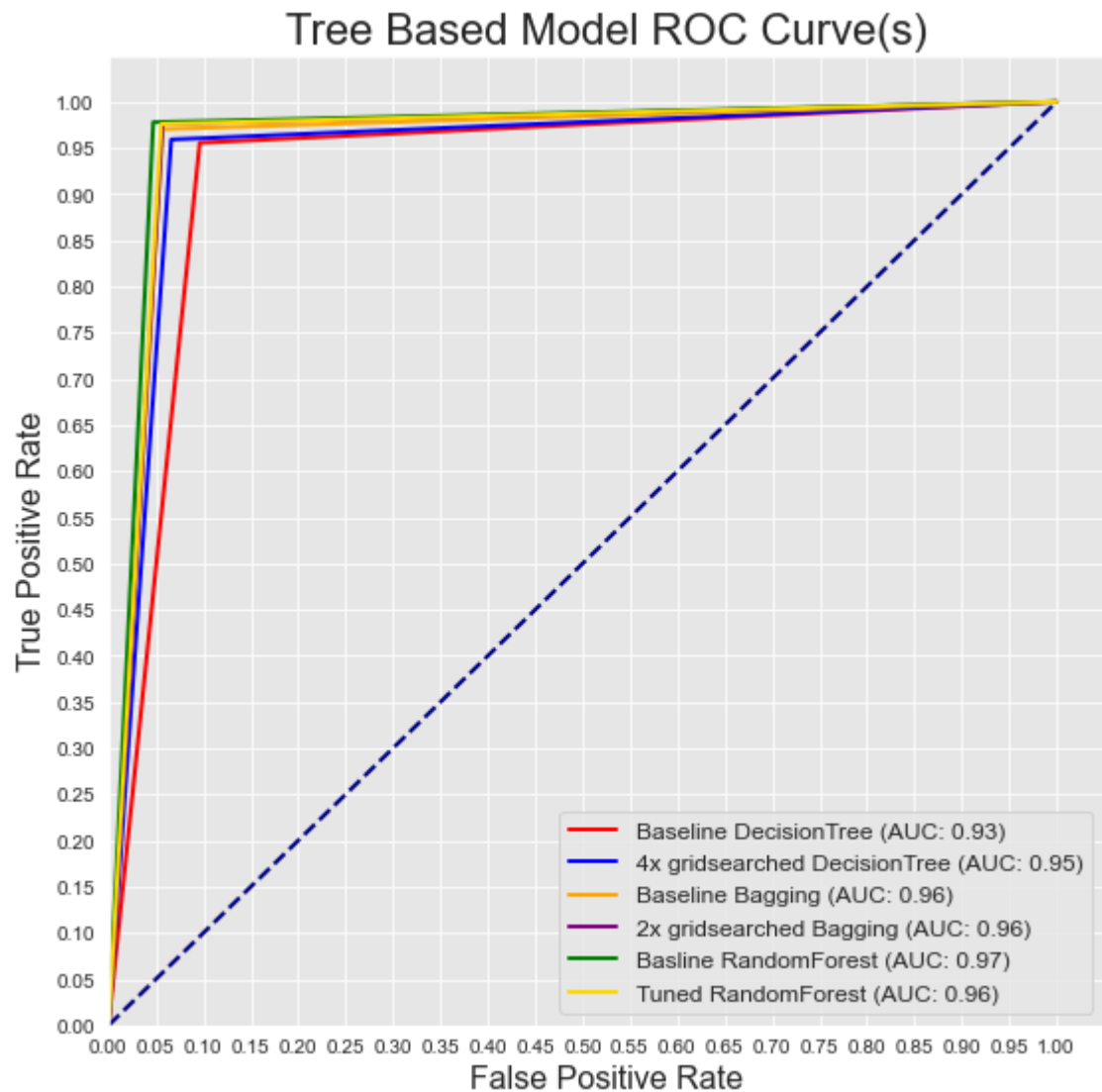
After selecting a resampling strategy, my next step was to discover the optimal tree-based model and related hyper parameters. I performed this investigation in the [gridsearching notebook](#) ([./development/gridsearching.ipynb](#)) found in the development/ folder of this repository, I discovered optimal hyperparameters using SciKit-Learn's [GridsearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html) function and searching different grids in an iterative manner. After fitting each gridsearch I evaluated the optimized model against the test data, focusing primarily -again- on F1-score and ROC AUC.

It turned out that the baseline Random Forest (sans hyperparameter tuning) was the strongest performer according to the F1-score. It should be noted that the computational expense of a gridsearch on a Random Forest on my machine was too great. So the Random Forest was not gridsearch but I experimented manually with some hyperparameters, using the best parameters found for the solo Decision Tree as a guideline. I found that any adjustments I made to the model only worsened its performance. The F1-score for the various tree-based models inspected is visualized below:

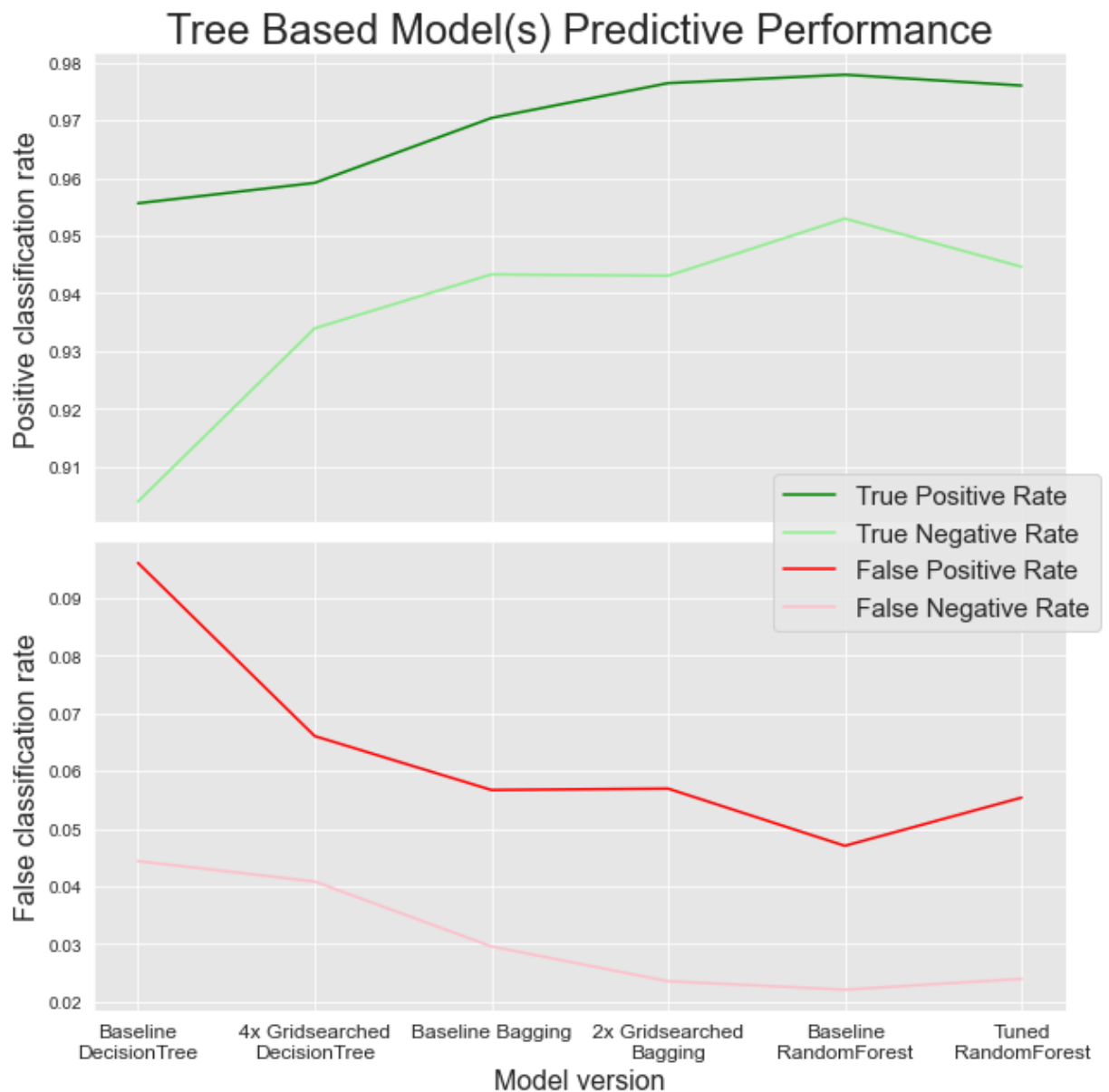
Tree Based Model(s) Performance



The curve of the ROC AUC scores was quite acute, which made me suspicious of overfitting... however the confusion matrices associated with each model gave me some peace of mind in that regard. It should be noted that the following ROC curves are of the model's performance with the test data. Nonetheless, the trainin data set alone is over 150 thousand records, so at this point the ROC curves should be taken with a grain of salt. However the visualization below certainly demonstrates the models' performance relative to one another at least.



As a final attempt at validating that the Random Forest, with its default hyperparameter values, is in fact the highest performing model, I plotted the rate of correct and incorrect predictions by each model. The following visualization essentially shows how each model's confusion matrix compares to the others. Even here the baseline Random Forest is the obvious best performer. Again, the following visualization represents each model's performance on the test data.



Final Model Performance Evaluation

In [12]:

```

1 # import dependencies
2 from sklearn.ensemble import RandomForestClassifier
3 from sklearn.metrics import classification_report, roc_curve, auc, confusion_matrix
4 from sklearn.model_selection import cross_val_score, cross_val_predict

```

In [13]:

```

1 # instantiate and fit model
2 RFC = RandomForestClassifier(n_jobs=3)
3 RFC.fit(X_resample, y_resample)

```

Out[13]: RandomForestClassifier(n_jobs=3)

Evaluate Training Data Classification Report and Confusion

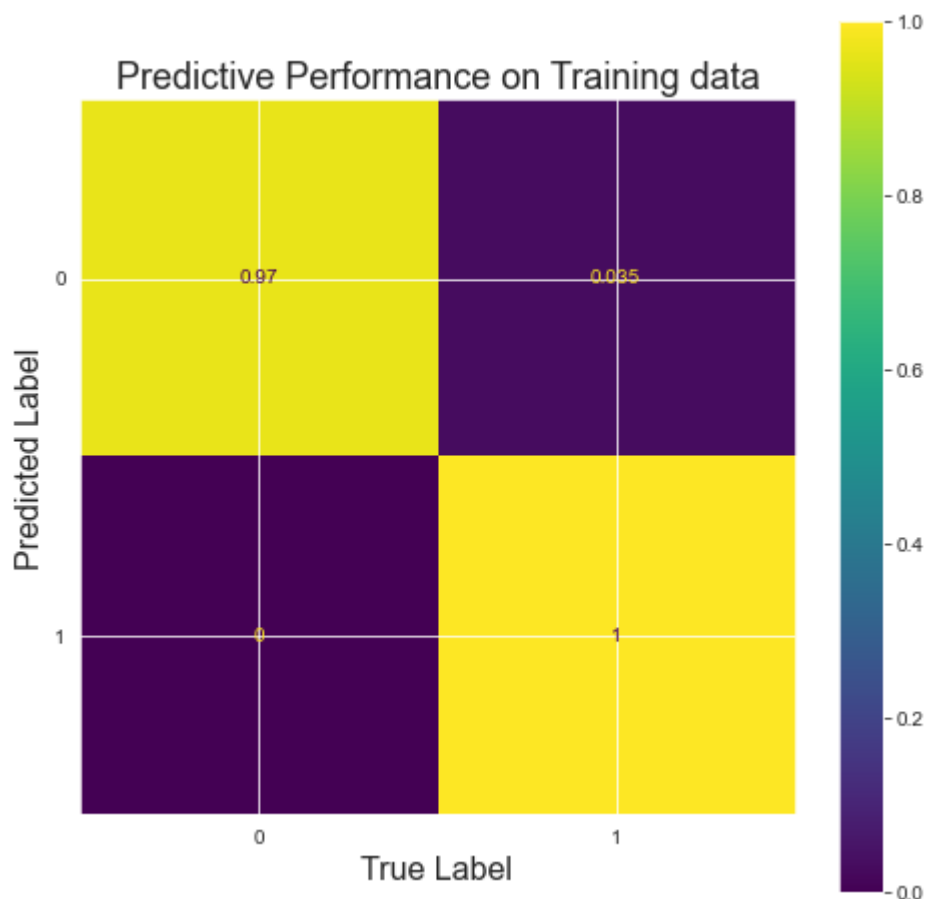
Evaluate Training Data Classification Report and Confusion Matrix

```
In [14]: 1 # evaluate on training predictions
2 train_pred = cross_val_predict(RFC,X_train,y_train,n_jobs=3)
3 rfc_train_report = classification_report(y_train,train_pred,output_dict=
4 rfc_train_report = pd.DataFrame(rfc_train_report).iloc[:,0:3]
5 rfc_train_report
```

```
Out[14]:
```

	0	1	accuracy
precision	0.986351	0.975572	0.984457
recall	0.994748	0.938412	0.984457
f1-score	0.990532	0.956631	0.984457
support	84923.000000	18981.000000	0.984457

```
In [15]: 1 fig, ax = plt.subplots(figsize=(8,8))
2 plot_confusion_matrix(RFC,X_train,y_train,normalize='true',ax=ax)
3 ax.set_title("Predictive Performance on Training data",fontsize=18)
4 ax.set_xlabel('True Label',fontsize=16)
5 ax.set_ylabel('Predicted Label',fontsize=16)
6
7 plt.show()
```



The data used on the report and matrix above is the training sample *before* resampling. I also use

the `cross_val_predict()` function in lieu of the more commonly used `.predict()` method to bolster the reliability of the predictions made. For this project I have been primarily concerned with F1-score which scores extremely high here, but it's worth noting that accuracy, precision and recall all individually score extremely high as well.

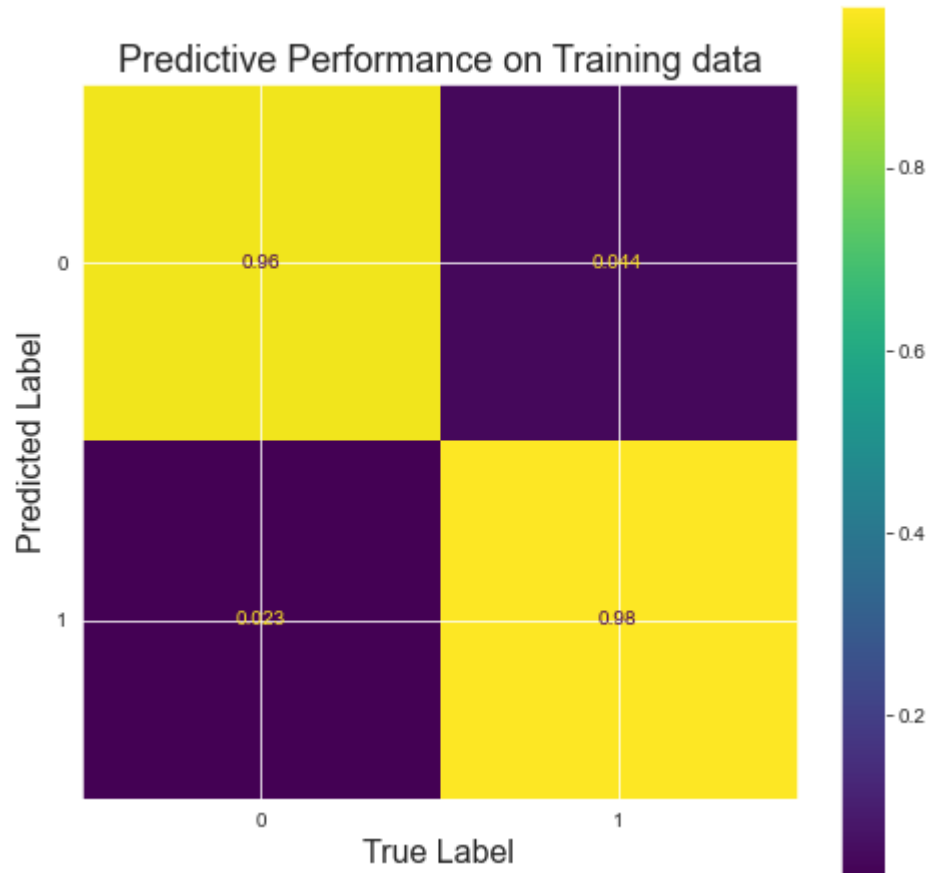
Evaluate Test Data Classification Report and Confusion Matrix

```
In [17]: 1 # evaluate on test data
2 test_pred = cross_val_predict(RFC,X_test,y_test,n_jobs=3)
3 rfc_test_report = classification_report(y_test,test_pred,output_dict='true')
4 rfc_test_report = pd.DataFrame(rfc_test_report).iloc[:,0:3]
5 rfc_test_report
```

Out[17]:

	0	1	accuracy
precision	0.980682	0.964796	0.977903
recall	0.992445	0.913732	0.977903
f1-score	0.986528	0.938570	0.977903
support	21177.000000	4799.000000	0.977903

```
In [18]: 1 fig, ax = plt.subplots(figsize=(8,8))
2 plot_confusion_matrix(RFC,X_test,y_test,normalize='true',ax=ax)
3 ax.set_title("Predictive Performance on Training data",fontsize=18)
4 ax.set_xlabel('True Label',fontsize=16)
5 ax.set_ylabel('Predicted Label',fontsize=16)
6
7 plt.show()
```



Test data results are extremely consistent with training data results. So this model is incredibly strong in the sense that it is compatible with a variety of classification tasks that may prioritize any one of the performance metrics (accuracy, F1-score, precision, recall)

Train vs Test ROC AUC

```
In [22]: ▶ 1 # Calculate the fpr, tpr, and thresholds for the training set
2 train_fpr , train_tpr, train_thresholds = roc_curve(y_train, train_pred)
3 train_auc = auc(train_fpr, train_tpr)
4 print('RFC train AUC: {}'.format(train_auc))
5
6 # Calculate the fpr, tpr, and thresholds for the testing set
7 test_fpr , test_tpr, test_thresholds = roc_curve(y_test, test_pred)
8 test_auc = auc(test_fpr, test_tpr)
9 print('RFC test AUC: {}'.format(test_auc))
```

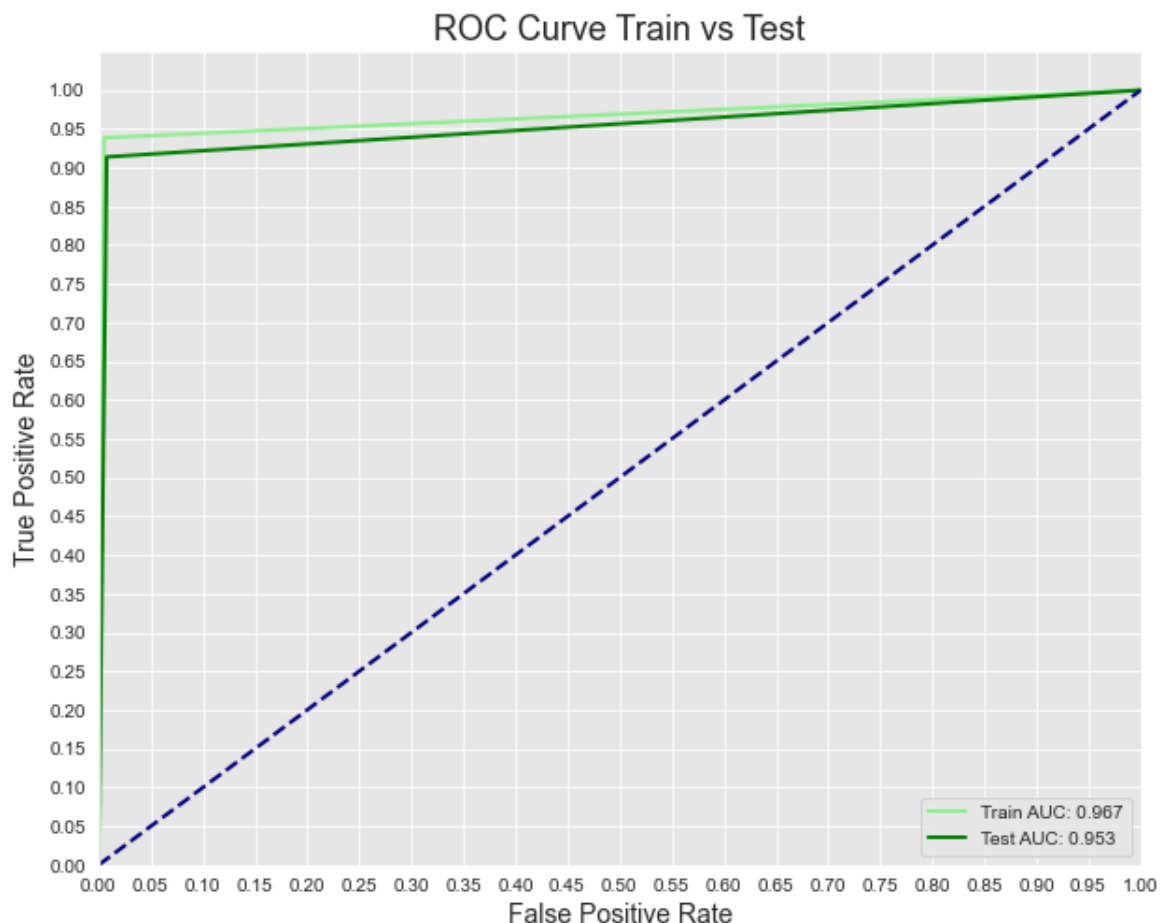
RFC train AUC: 0.9665801399777751

RFC test AUC: 0.9530883304171708

```

In [23]: ▶ 1 plt.figure(figsize=(10,8))
2 lw = 2
3
4 plt.plot(train_fpr, train_tpr, color='lightgreen',
5          lw=lw, label=f'Train AUC: {round(train_auc,3)}')
6 plt.plot(test_fpr, test_tpr, color='green',
7          lw=lw, label=f'Test AUC: {round(test_auc,3)}')
8
9 # Formatting
10 plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
11 plt.xlim([0.0, 1.0])
12 plt.ylim([0.0, 1.05])
13 plt.yticks([i/20.0 for i in range(21)])
14 plt.xticks([i/20.0 for i in range(21)])
15 plt.xlabel('False Positive Rate', fontsize=14)
16 plt.ylabel('True Positive Rate', fontsize=14)
17 plt.title('ROC Curve Train vs Test', fontsize=18)
18 plt.legend(loc="lower right")
19 plt.show()

```



ROC curves are known to be somewhat unreliable with large amounts of data like this project utilizes, nonetheless I felt it was worth plotting. Not much being told here that we don't already know from the classification reports and confusion matrices.

Model Interpretation and Business

Recomendations

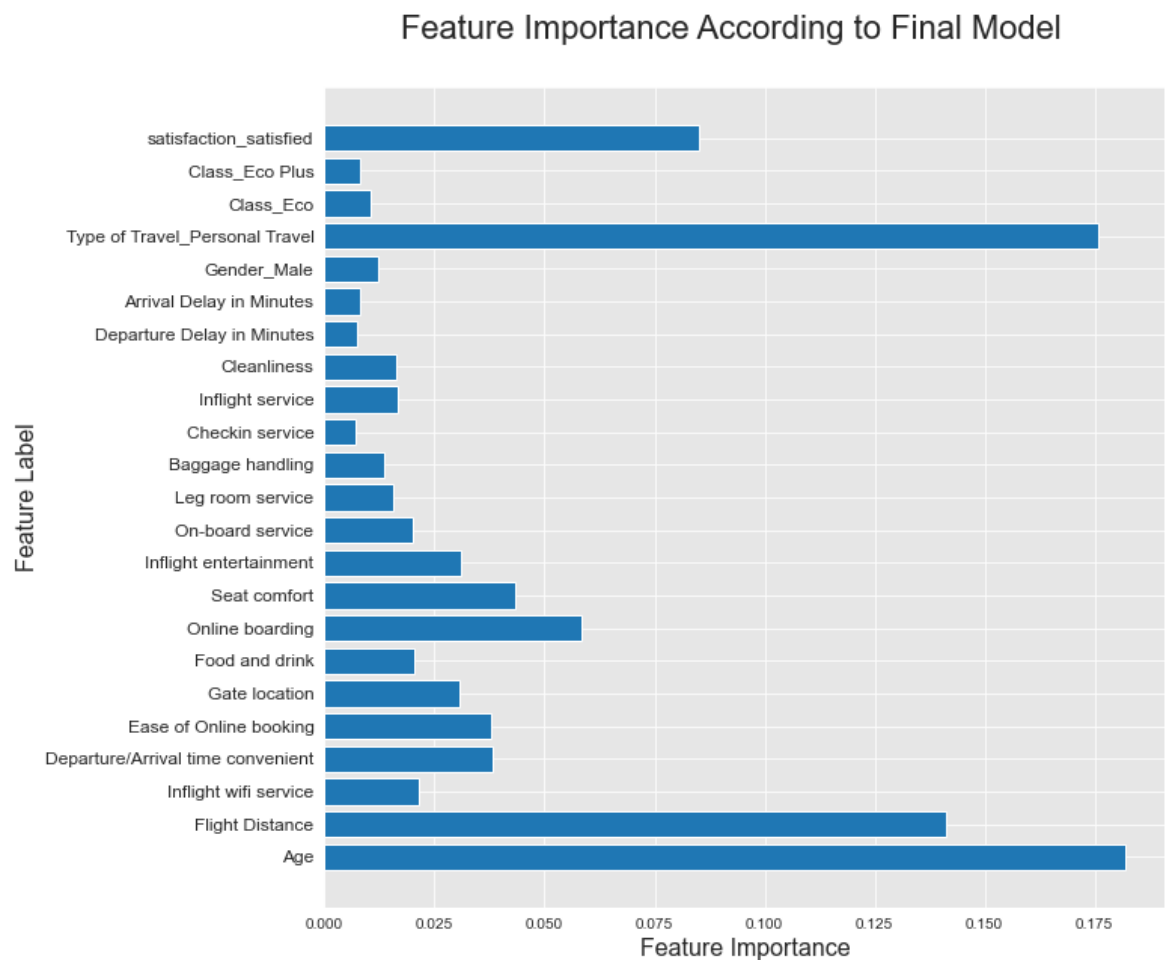
I conducted a thorough descriptive analysis of the data based on the model's feature importance rankings. The full analysis is annotated in the final section of the [gridsearch notebook](#) ([./development/gridsearching.ipynb](#)), all the code used to render plots and massage the data into plottable formates is available there as well. Therefore, here I will be linking the graphs from that notebook and sumarizing my analysis and discoveries.

Feature Importance


```

In [25]: 1 def plot_feature_importances(model):
2         n_features = X_train.shape[1]
3         plt.figure(figsize=(10,10))
4         plt.barh(range(n_features), model.feature_importances_, align='center')
5         plt.yticks(np.arange(n_features), X_train.columns.values, fontsize=12)
6         plt.xlabel('Feature importance')
7         plt.ylabel('Feature')
8
9         plot_feature_importances(RFC)
10        plt.suptitle("\nFeature Importance According to Final Model", fontsize=22)
11        plt.xlabel('Feature Importance', fontsize=16)
12        plt.ylabel('Feature Label', fontsize=16)
13        plt.show()

```



Instead of linking the .png file I actually plotted the feature importances of the model object in this notebook, just to validate that the model is consistent accross instances in other notebooks. The plot above is identical to the one in the original gridsearching notebook.

Obviously Age of the customer, their purpose for travel (business/personal), the overall satisfaction of the customer (satisfied/disatisfied), as well as the distance of the flight were the features the model relies on the most when making a prediction on the loyalty of the customer.

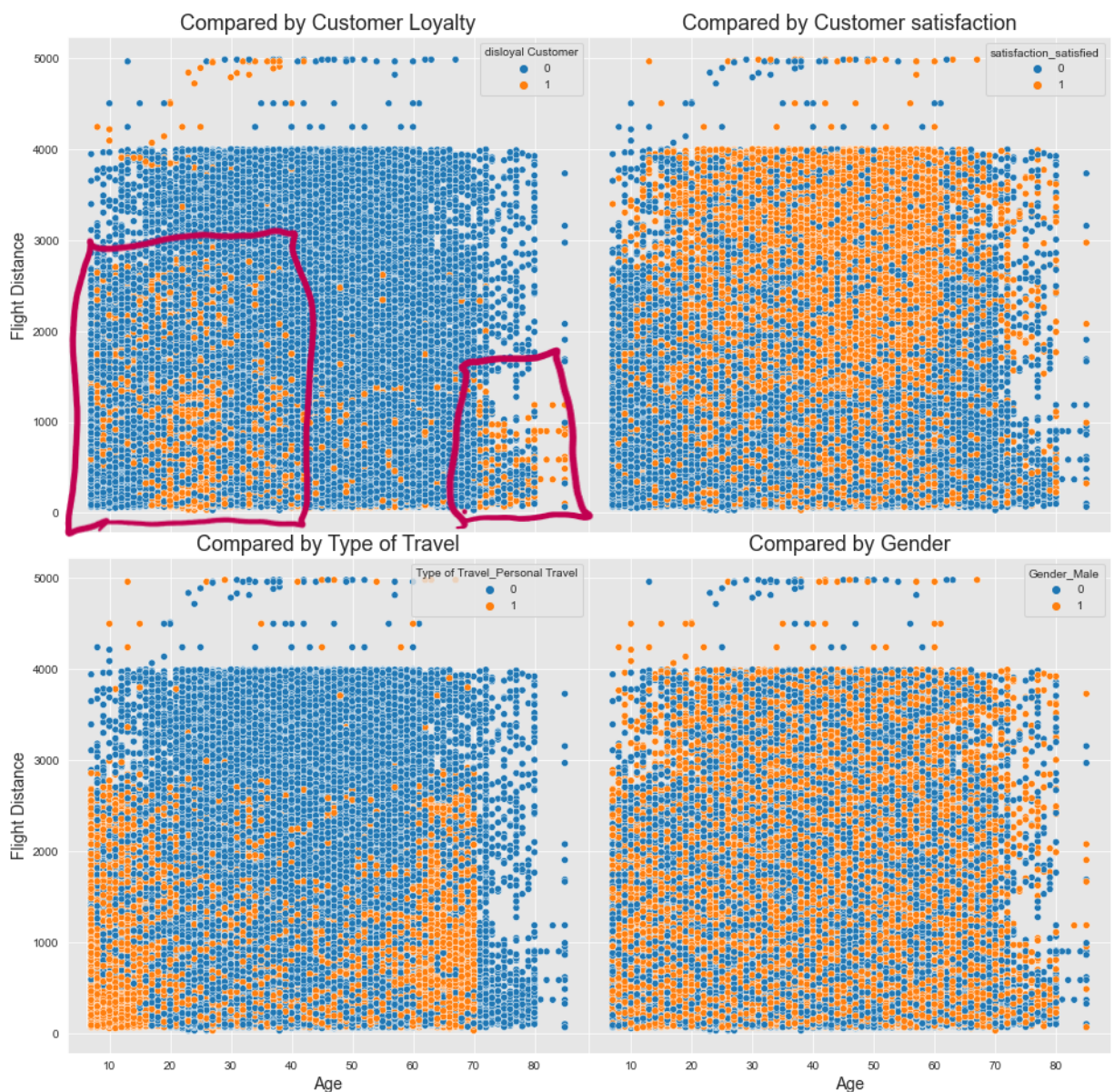
Interstingly the class of ticket (economy/business/economy plus), as well as arrival/departure delays, and check-in service are the least important factors to the model.

Looking strictly at the survey response features, the most important are the online services relates to check-in (online boarding, online booking), factors regarding the airport itself and the itinerary (parture/arival time convenience and gate location), and factors regarding inflight comfort (wifi, leg room, food/drink and entertainment). Those aforementioned features will be the focus of the descriptive analysis to discover patterns that may inform us as to what aspects of the customer experience of disloyal customers needs to be improved the most.

Age vs Flight Distance vs Boolean Features

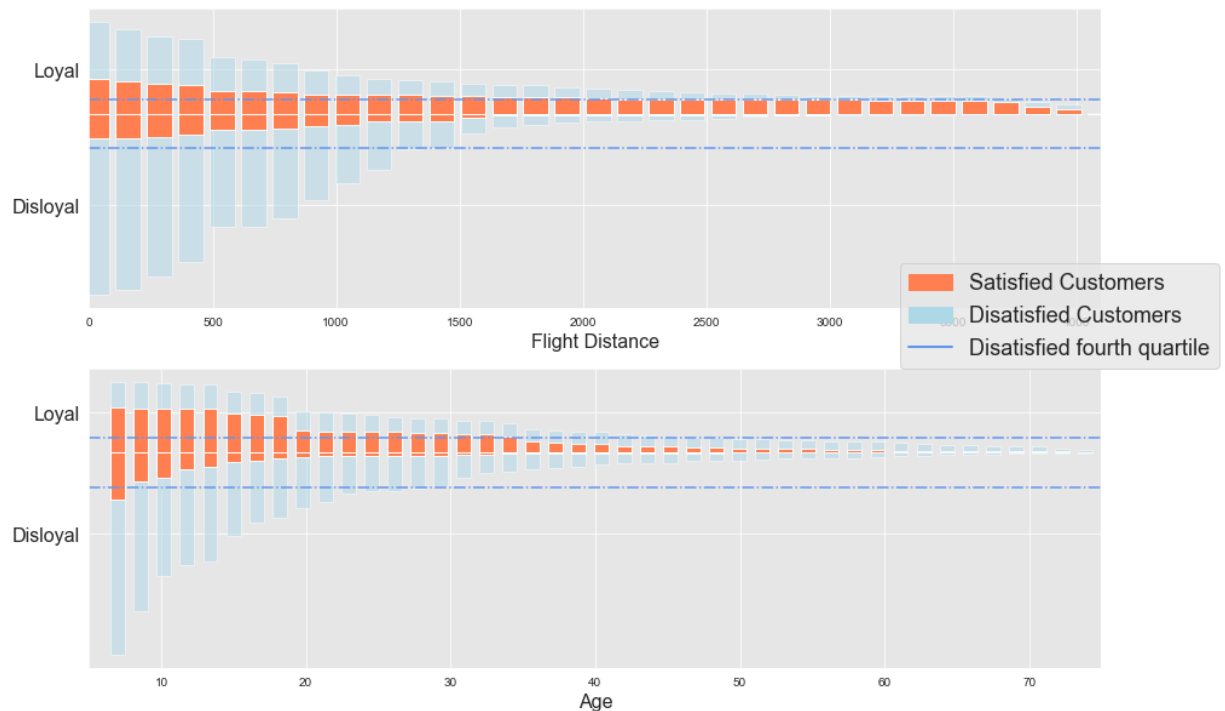
My first step after evaluating the feature importances was to cross reference the Age and Flight Distance with the other most important boolean features. Note that gender here is plotted mostly just to fill in what would be an awkward blank square. I decided to focus my analysis on the clusters outlined in red. Because these appear to be the areas with the greatest density of disloyal customers (the target of this analysis).

Age vs Flight Distance



The scatter plots above gave me a clue as to what clusters of the data to focus on, but it didn't give me a hint as to how flight distance and/or age relates to customer satisfaction nor how that might differ between loyal and disloyal customers. So I made the following bar charts to discover any such relationships.

Customer Satisfaction By Age, Flight Distance and Loyalty



For both age and flight distance the upper quartile of dissatisfaction (in terms of volume) constitutes roughly the lower half of the range of the respective feature. In other words the most dissatisfaction is found in the first and second quartiles of age and flight distance. Some of this may be due to relative scarcity of data for the more extreme ranges. Unsiprisingly, disloyal customers are the most dissatisfied, especially at the lower ranges of age and flight distance.

It is unlikely that improving satisfaction for flyers under 16 would have any real impact on loyalty conversion save for their satisfaction being a proxy to the parents satisfaction, which can not be made clear from this data.

It is clear however that there is the bulk of dissatisfaction to be addressed can be found on flights udner ~2500 miles and individuals under ~45.

Satisfaction, a boolean featurer where 1 represents an overall satisfied customer, is one of the most important features for the model, however it is certainly a proxy to the aggregate satisfaction of a customers survey responses, some of which were fairly important themselves. So I built the following plot in order to examine the relationship between overall customer satisfaction as a boolean value and aggregate customer satisfaction for each survey question as an ordinal categorical value (1-5).



Again, unsurprisingly there is far more dissatisfaction found within the disloyal sector of the market. Although there is more dissatisfaction by volum from disloyal customers, the degree of dissatisfaction for one survey question relative to the others is fairly consistent between loyal and disloyal customers.

Baggage handling and inflight wifi service share the greatest portion of dissatisfaction. Baggage handling was not an outstandingly important feature to the model, however inflight wifi service was fairly important. Leg-room, another fairly important feature also shares a large portion of the dissatisfaction held by the disloyal customers. Food and drink as well as inflight entertainment deserve attention here as well.

I also wanted to evaluate customer satisfaction per survey response constrained to the clusters outlined in the scatter plot above, as well as for business travel since it generally overlaps with those age/flight distance clusters.

fig. 1

fig. 2



In the above visualization, fig. 1 on the left shows the market share of each group represented, and fig. 2 on the right shows that groups aggregate survey responses (i.e. each feature is summed, bar height represents sum total of responses). Red represents the bottom quartile of response for that group, green represents the fourth quartile, and yellow represents the second and third quartiles.

Judging by market share alone, efforts to improve satisfaction to senior citizens would have little impact, if even noticeable, its advisable to focus efforts on the groups represented by the top and bottom charts here.

Of the survey questions alone, online boarding is the most important feature. Business travelers as well as young adults scored this feature in the first quartile of responses (extremely poorly) and seniors scored it still no higher than third quartile.

Inflight wifi, and ease of online booking were also extremely important to the model, and customers accross the board scored these features in the bottom quartile, along with departure/arrival times which were also relatively important to the model.

Business travelers and young adults are generally not thrilled with the food and drink, seat comfort, cleanliness, and inflight entertainment, as well as gate location.

Final Recommendations

I recommend focusing efforts to improve satisfaction of customer experience for individuals traveling for business purposes and/or between the ages of 16 and 40, for flights 3000 miles or less.

Specific aspects of the customer experience on those flights that I predict will have the greatest return on efforts made are the customer's online experience (online booking, online boarding), internet dependent in-flight services such as wifi, and entertainment, and comfort related features of the flight itself such as seat comfort, food and drink, and overall cleanliness.

Improving gate location as well as the convenience of the departure and arrival times would have a great impact according to my analysis of the data, however I do not believe it would be economical to implement changes here because it would require significant coordination and concession by and width other airlines as well as the FAA.

It is best to focus on improving inflight services like wifi and entertainment, refreshments and seat comfort, and the UX of pre-boarding online services for business flights and youn adults flying less than 3000 miles.

Type *Markdown* and LaTeX: α^2