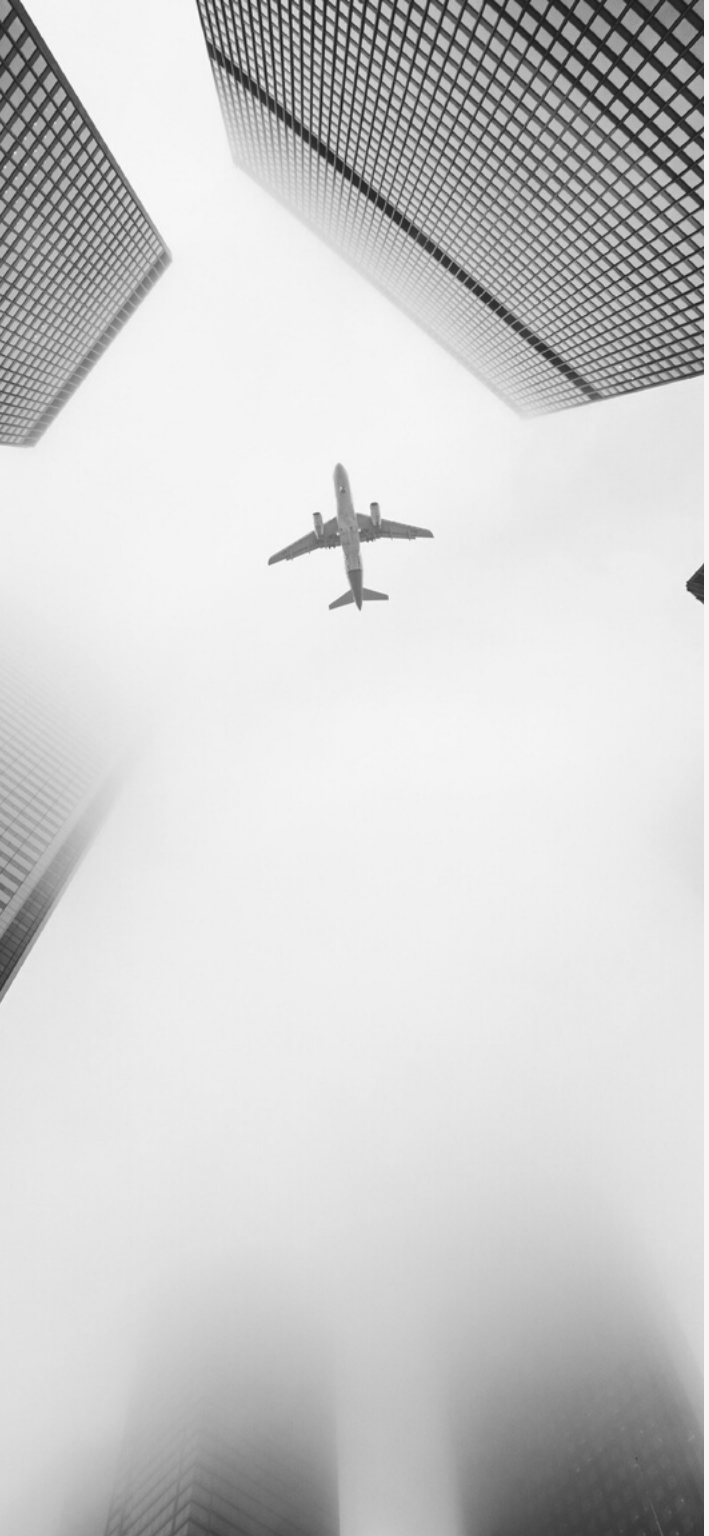
01

ML · PROJECT

Flight Passenger Satisfaction

Anthony . Benny . Venus

→



PROJECT OBJECTIVE

Intended to understand passenger satisfaction by analyzing 22 factors.

- Customer demographic data
- In-flight experiences
- Flight information

→

03

FOCUS

FIRST QUESTION

Can passenger satisfaction be anticipated?

SECOND QUESTION

What factors are associated with customer satisfaction?

→

CUSTOMER DEMOGRAPHIC DATA

1. Age
2. Gender: Female, Male
3. Customer Type: Loyal customer, Disloyal customer
4. Type of Travel: Personal Travel, Business Travel

IN-FLIGHT EXPERIENCES

(*Satisfaction level: 0 = Not Applicable, 1-5)

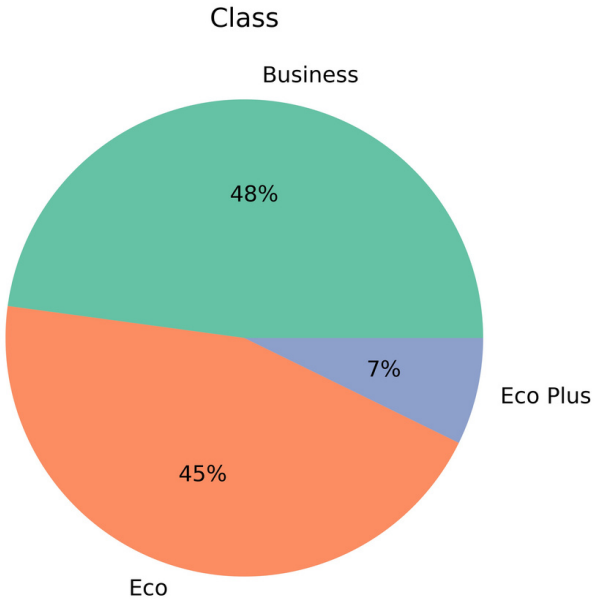
5. Inflight wifi service*
6. Food and drink*
7. Seat comfort*
8. Inflight entertainment*
9. Leg room service*
10. Inflight service*
11. Cleanliness*

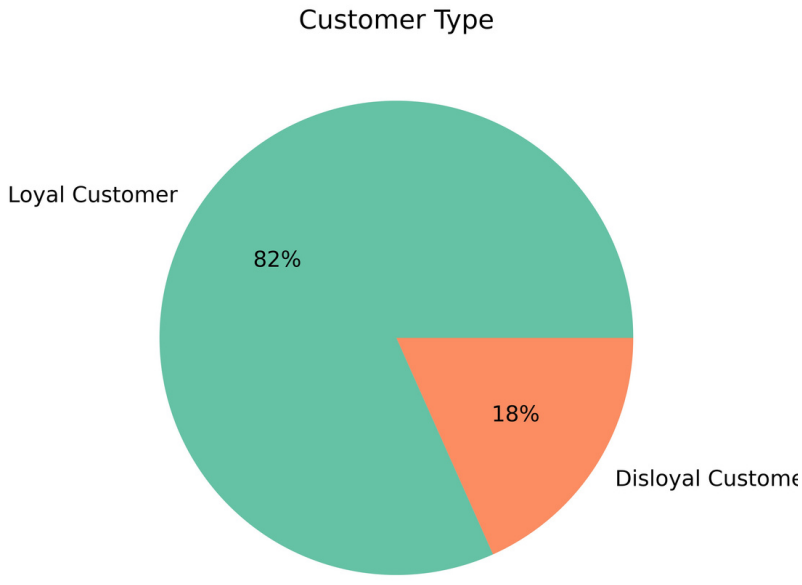
FLIGHT INFORMATION & EXPERIENCES

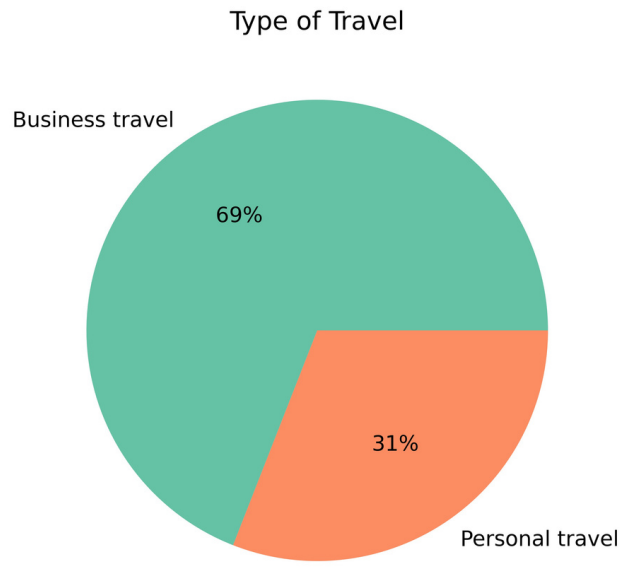
(*Satisfaction level: 0 = Not Applicable, 1-5)

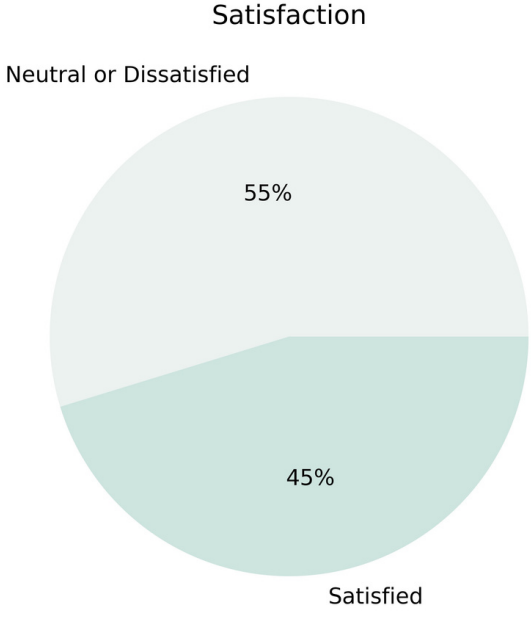
12. Class: Business, Eco, Eco Plus
13. Flight distance
14. Departure/Arrival time convenient*
15. Ease of Online booking*
16. Gate location*
17. Online boarding*
18. On-board service*
19. Baggage handling*
20. Check-in service*
21. Departure Delay in Minutes
22. Arrival Delay in Minutes

EXPLORATORY DATA ANALYSIS (EDA)









- The data set contains 129,880 responses from passengers aged 7-85 with 82% loyal customers.
- Business travel is the majority of flight purposes. However, only 48% of passengers purchased business class tickets.
- The distribution of passenger satisfaction was fairly balanced. Nearly half of the customers expressed satisfied and the rest were neutral to dissatisfied.

IMPORT LIBRARIES

- pandas
- numpy
- matplotlib. pyplot
- seaborn
- %matplotlib inline
- sklearn. model_selection. train_test_split
- sklearn. preprocessing. standarscaler
- imblearn. over_sampling. SMOTE
- imblearn. combine. SMOTEENN
- sklearn. neighbors. KNeighborsClassifier
- sklearn. metrics. confusion_matrix
- sklearn. metrics. classification_report
- sklearn. metrics. accuracy_score
- sklearn. tree. DecisionTreeClassifier
- sklearn. ensemble. RandomForestClassifier

IMPORT LIBRARIES

- sklearn. naive_bayes. GaussianNB
- sklearn. svm. SVC
- sklearn. linear_model. LogisticRegression
- sklearn. model_selection
- sklearn. ensemble. BaggingClassifer
- sklearn. ensemble. ExtraTreesClassifier
- sklearn. ensemble. AdaBoostClassifier
- sklearn. ensemble. GradientBoostingClassifier
- sklearn. ensemble. VotingClassifier
- sklearn. decomposition. PCA
- sklearn. discriminant_analysis. LinearDiscriminantAnalysis
- sklearn.model_selection. RandomizedSearchCV
- sklearn. model_selection. GridSearchCV

PRE-PROCESSING


```
df.isna().sum()   #find out that there are 393 null values in the col. "Arrival Delay in Minutes"
```

```
id                                    0
satisfaction_v2                       0
Gender                                0
Customer Type                         0
Age                                   0
Type of Travel                        0
Class                                 0
Flight Distance                       0
Seat comfort                          0
Departure/Arrival time convenient     0
Food and drink                        0
Gate location                         0
Inflight wifi service                 0
Inflight entertainment                0
Online support                        0
Ease of Online booking                0
On-board service                      0
Leg room service                      0
Baggage handling                      0
Checkin service                       0
Cleanliness                           0
Online boarding                       0
Departure Delay in Minutes            0
Arrival Delay in Minutes            393
dtype: int64
```

CHECK DATA

FILL IN MISSING DATA

```
df.fillna(value=df['Arrival Delay in Minutes'].mean(),inplace = True)
```

ONE HOT ENCODING

```
df["one_hot_Gender"] = df["Gender"].apply(lambda x: 1 if x == "Male" else 0)

df["one_hot_Customer Type"] = df["Customer Type"].apply(lambda x: 1 if x == "Loyal Customer" else 0)

df["one_hot_Type of Travel"] = df["Type of Travel"].apply(lambda x: 1 if x == "Personal Travel" else 0)

df["one_hot_Class_E"] = df["Class"].apply(lambda x: 1 if x == "Eco" else 0)
df["one_hot_Class_EP"] = df["Class"].apply(lambda x: 1 if x == "Eco Plus" else 0)
df["one_hot_Class_B"] = df["Class"].apply(lambda x: 1 if x == "Business" else 0)
```

→

MODEL CREATION

SPLIT THE DATA

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```

FEATURE SCALING

Standardize data

```
[15] # standardize data
     from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.fit_transform(X_test)
```

MODEL CREATION

IMBALANCE DATA

Smote

```
[16] from imblearn.over_sampling import SMOTE
     sm_ = SMOTE()
     X_train_sen, y_train_sen = sm_.fit_resample(X_train, y_train)
```

Smoteenn

```
[16] from imblearn.combine import SMOTEENN
     sm = SMOTEENN()
     X_train_sen, y_train_sen = sm.fit_resample(X_train, y_train)
```

MODEL CATALOGUE

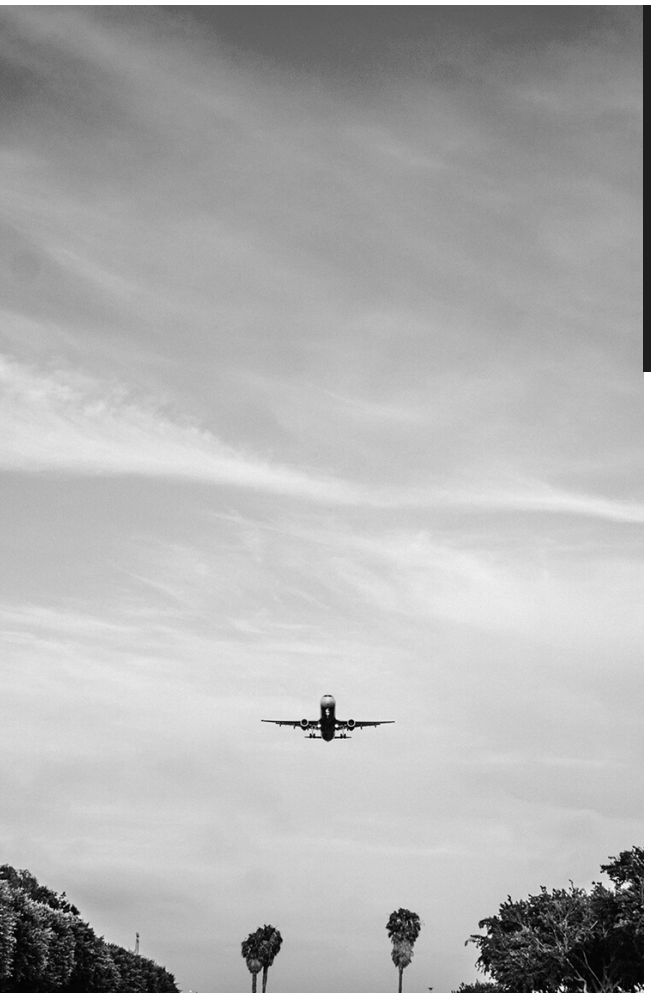

```python
# start trying different classifier models
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
```

```python
model_kNN = KNeighborsClassifier()
model_dt = DecisionTreeClassifier(criterion='entropy')
model_rf = RandomForestClassifier(n_estimators=200)
model_nb = GaussianNB()
model_sv = SVC()
model_lr = LogisticRegression()

model_list = [model_kNN,model_dt,model_rf,model_nb,model_sv,model_lr]
```

→

GET RESULTS


```
8 from sklearn.metrics import classification_report, accuracy_score
```



```
1 for model in model_list:
2   model.fit(X_train, y_train)
3   y_pred = model.predict(X_test)
4   result = accuracy_score(y_test, y_pred)
5   print(model,'\n','Accuracy is: ', result*100,'\n')
6   print(classification_report(y_test, y_pred))
7 # test size: 0.25
```

Results:

- classification report
- accuracy score

For-loop to generate results for all models in bulk.

→

RESULTS


```
KNeighborsClassifier()
 Accuracy is:  92.68247613181398

              precision    recall  f1-score   support

           0       0.90      0.94      0.92     14688
           1       0.95      0.92      0.93     17782

    accuracy                           0.93     32470
   macro avg       0.93      0.93      0.93     32470
weighted avg       0.93      0.93      0.93     32470
```

```
DecisionTreeClassifier(criterion='entropy')
 Accuracy is:  93.96673852787188

              precision    recall  f1-score   support

           0       0.93      0.93      0.93     14688
           1       0.94      0.95      0.94     17782

    accuracy                           0.94     32470
   macro avg       0.94      0.94      0.94     32470
weighted avg       0.94      0.94      0.94     32470
```

```
RandomForestClassifier(n_estimators=200)
 Accuracy is:  95.8484755158608

              precision    recall  f1-score   support

           0       0.95      0.96      0.95     14688
           1       0.97      0.95      0.96     17782

    accuracy                           0.96     32470
   macro avg       0.96      0.96      0.96     32470
weighted avg       0.96      0.96      0.96     32470
```

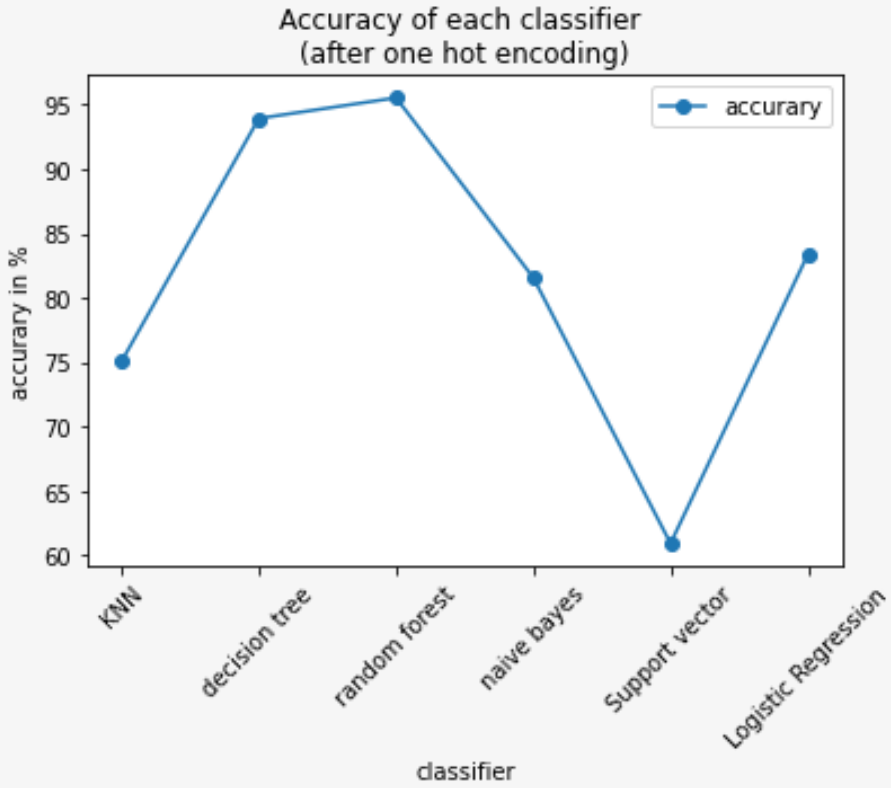
```
GaussianNB()
 Accuracy is:  81.71850939328611

              precision    recall  f1-score   support

           0       0.80      0.79      0.80     14688
           1       0.83      0.84      0.83     17782

    accuracy                           0.82     32470
   macro avg       0.82      0.82      0.82     32470
weighted avg       0.82      0.82      0.82     32470
```

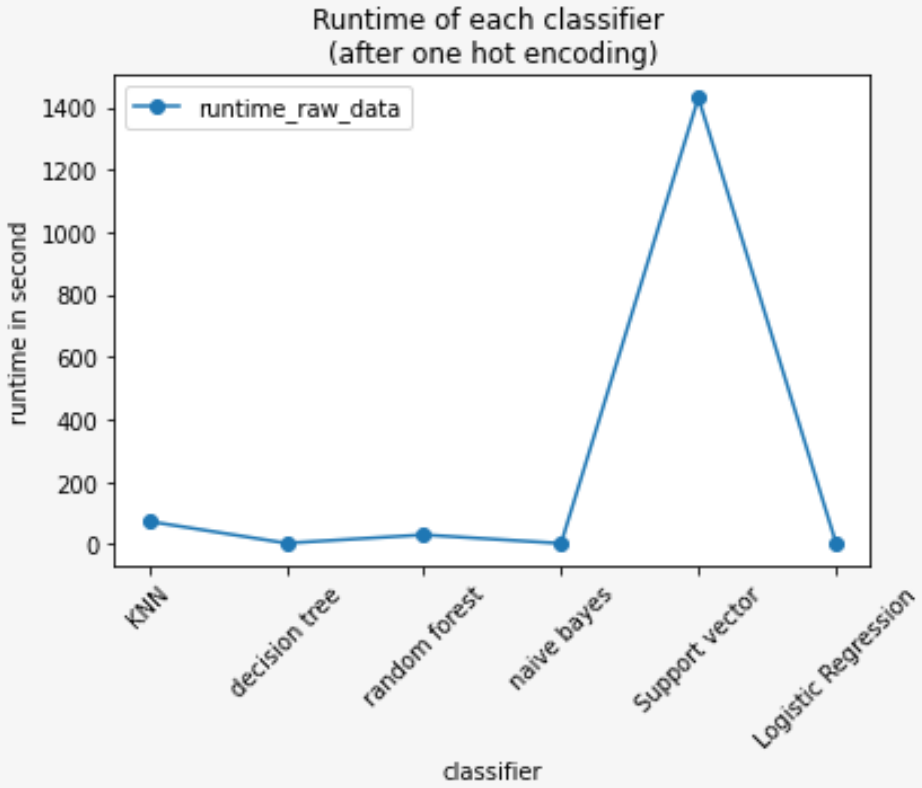
```
SVC()
 Accuracy is:  94.15768401601478

              precision    recall  f1-score   support

           0       0.94      0.93      0.94     14688
           1       0.94      0.95      0.95     17782

    accuracy                           0.94     32470
   macro avg       0.94      0.94      0.94     32470
weighted avg       0.94      0.94      0.94     32470
```

```
LogisticRegression()
 Accuracy is:  83.47397597782569

              precision    recall  f1-score   support

           0       0.82      0.82      0.82     14688
           1       0.85      0.85      0.85     17782

    accuracy                           0.83     32470
   macro avg       0.83      0.83      0.83     32470
weighted avg       0.83      0.83      0.83     32470
```

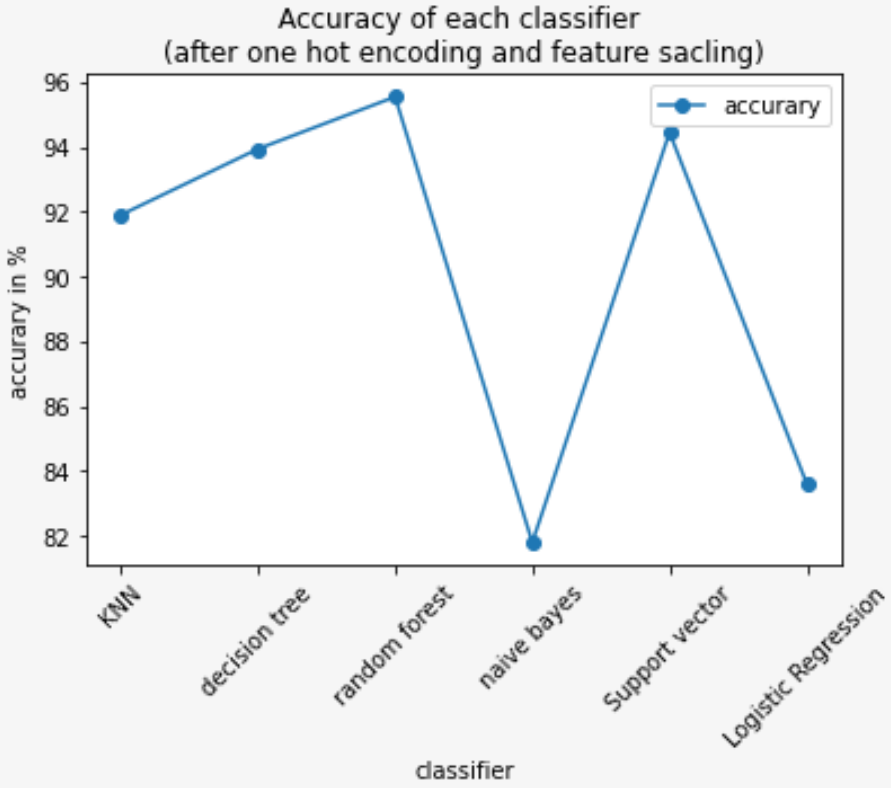
Winner: Random Forrest (95.85% accuracy)

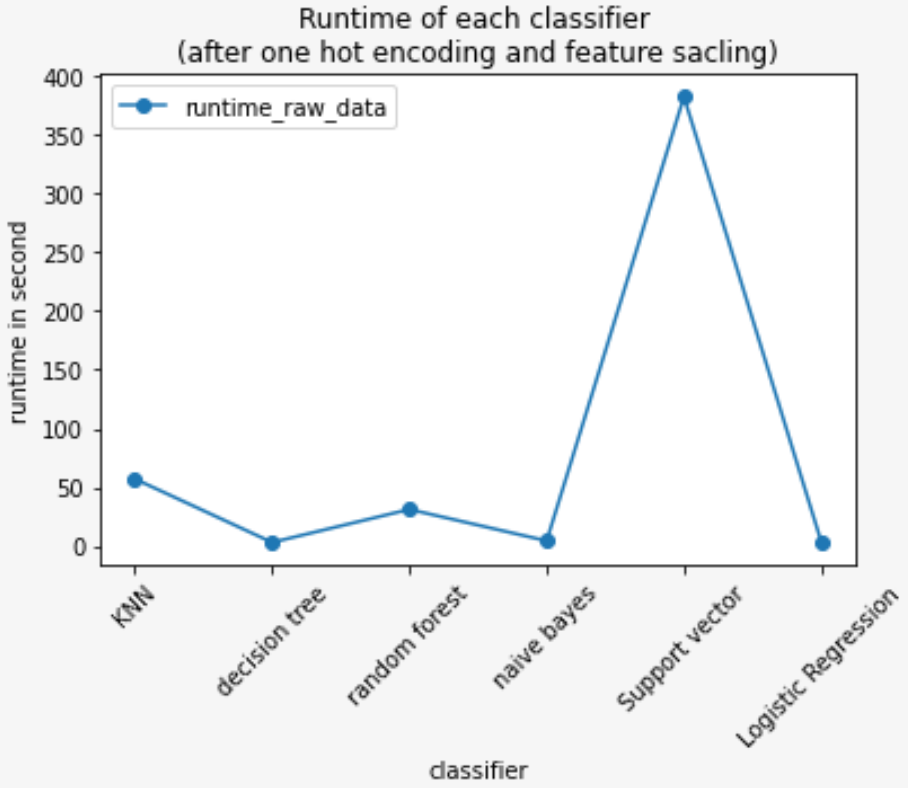
CLASSIFIER USAGE FEASIBILITY



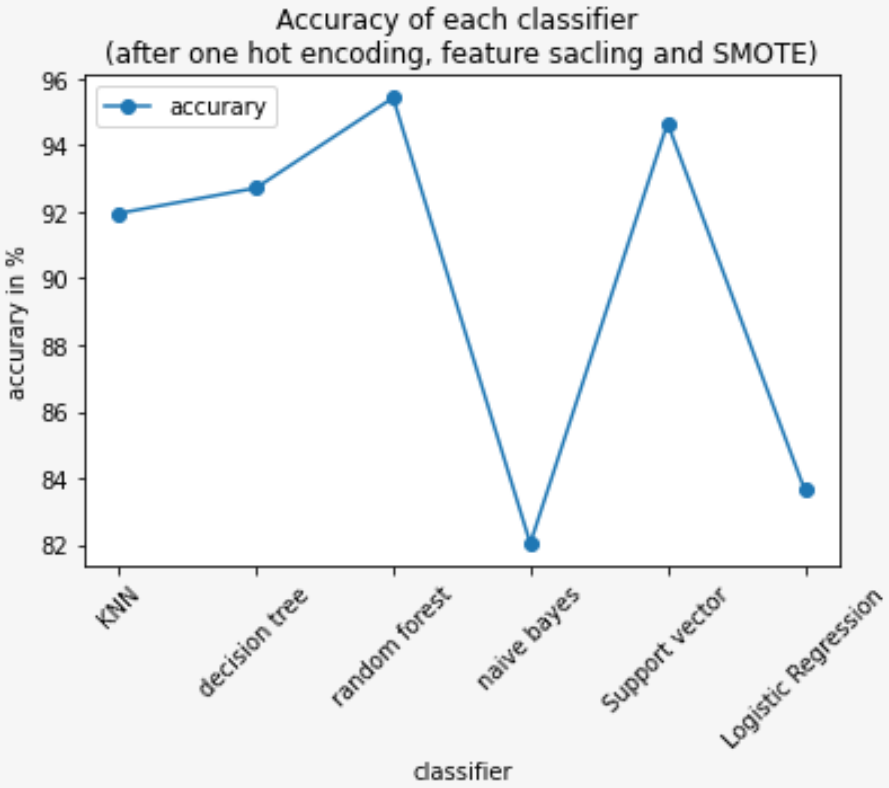


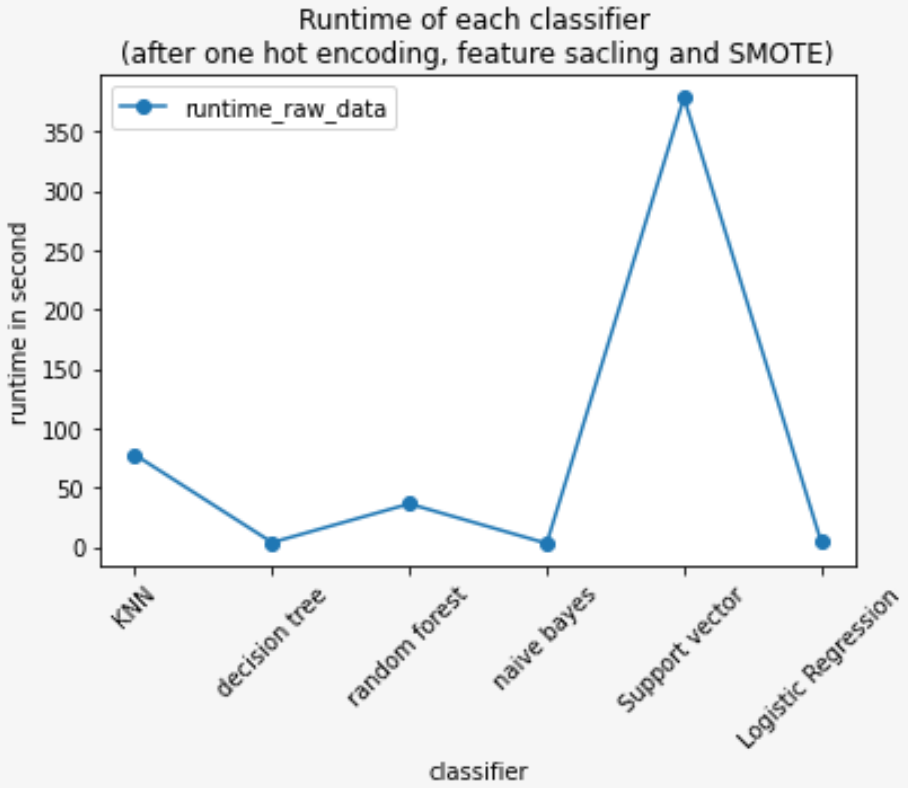
CLASSIFIER USAGE FEASIBILITY



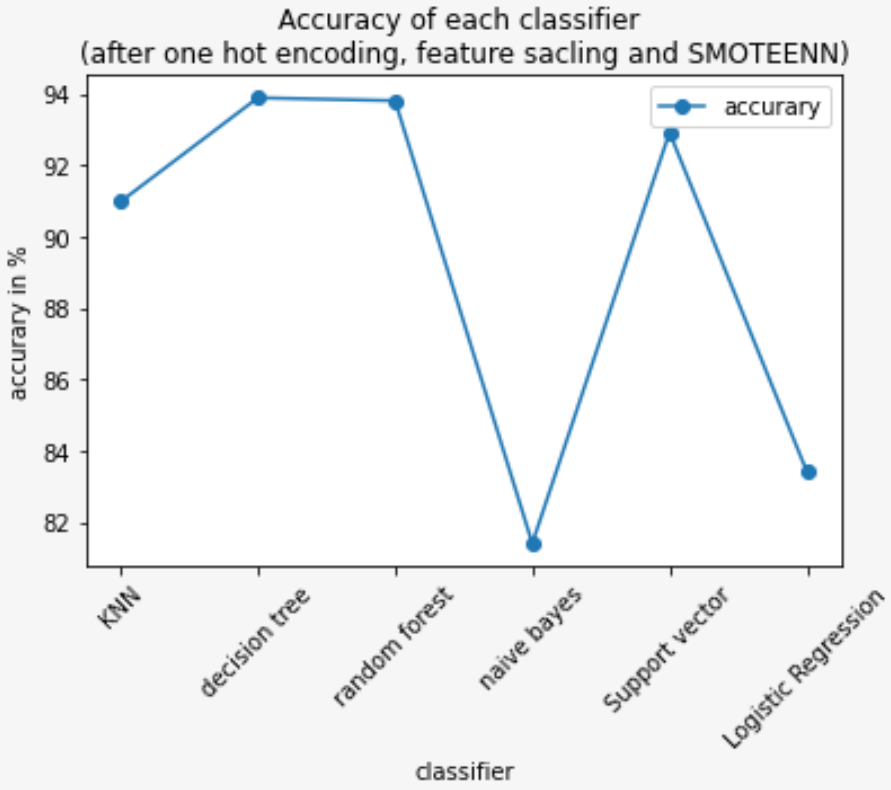


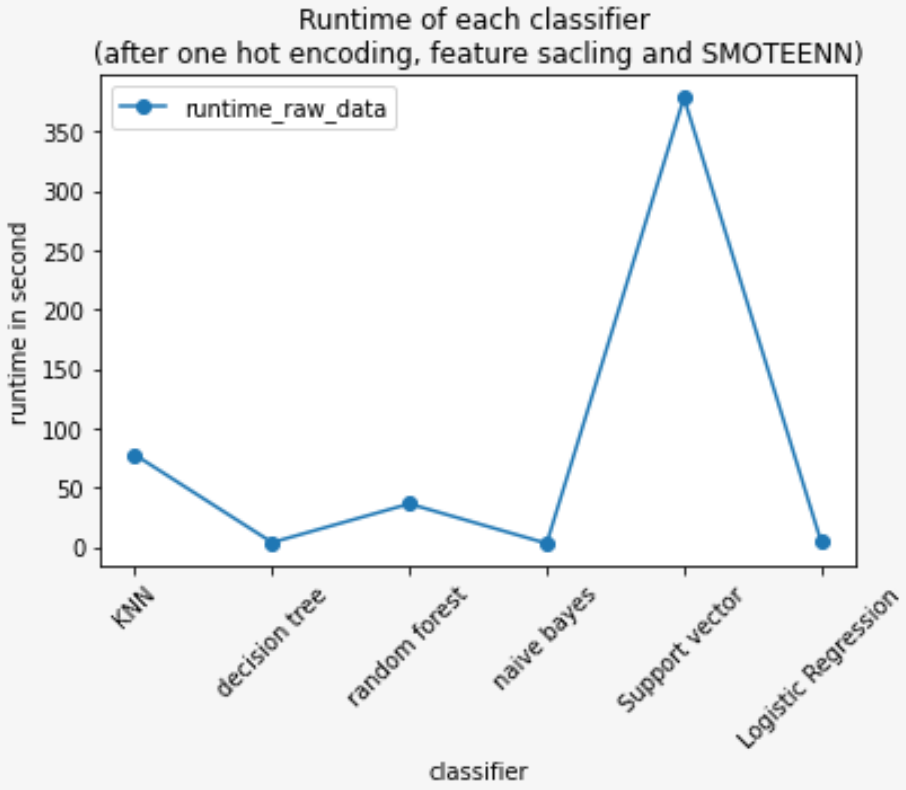
CLASSIFIER USAGE FEASIBILITY



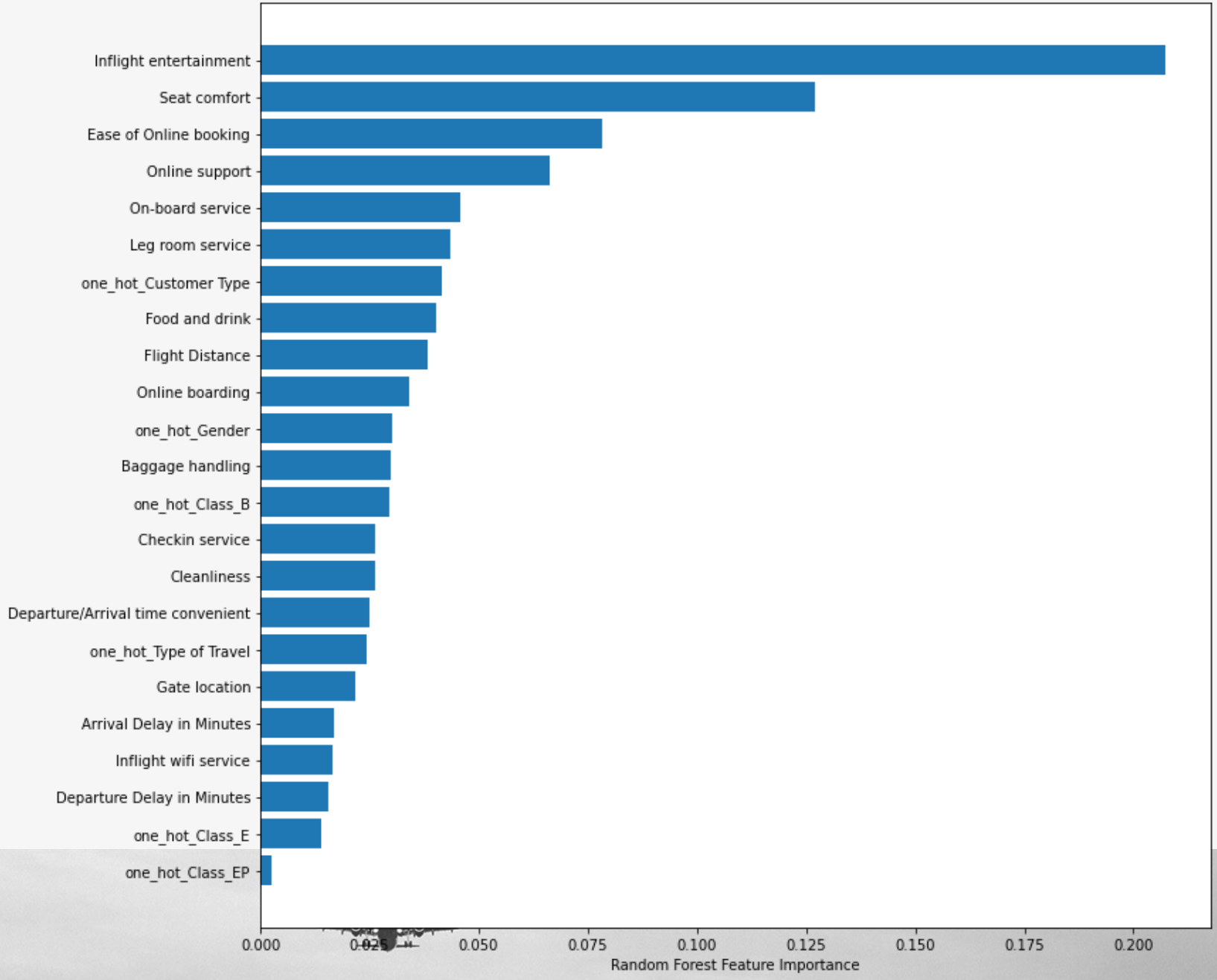


CLASSIFIER USAGE FEASIBILITY





RANDOM FOREST FEATURE IMPORTANCE



PCA

So far we have our winner, Random Forrest, with 95.85% accuracy. We then tried to find out the min. no. of features each model needs for >80% accuracy by PCA process.

```
model_list = [model_kNN,model_dt,model_rf,model_nb,model_sv,model_lr]
```

```
#Applying PCA
from sklearn.decomposition import PCA
```

```
com_list = list(range(5,22,5))
result_list = []

for n_com in com_list:
  pca = PCA(n_components = n_com)
  X_train_pca = pca.fit_transform(X_train)
  X_test_pca = pca.transform(X_test)
  for model in model_list:
    model.fit(X_train_pca, y_train)
    y_pred = model.predict(X_test_pca)
    result = accuracy_score(y_test, y_pred)
    r = []
    r.append(n_com)
    r.append(model)
    r.append(result*100)
    result_list.append(r)

result_list
```

PCA

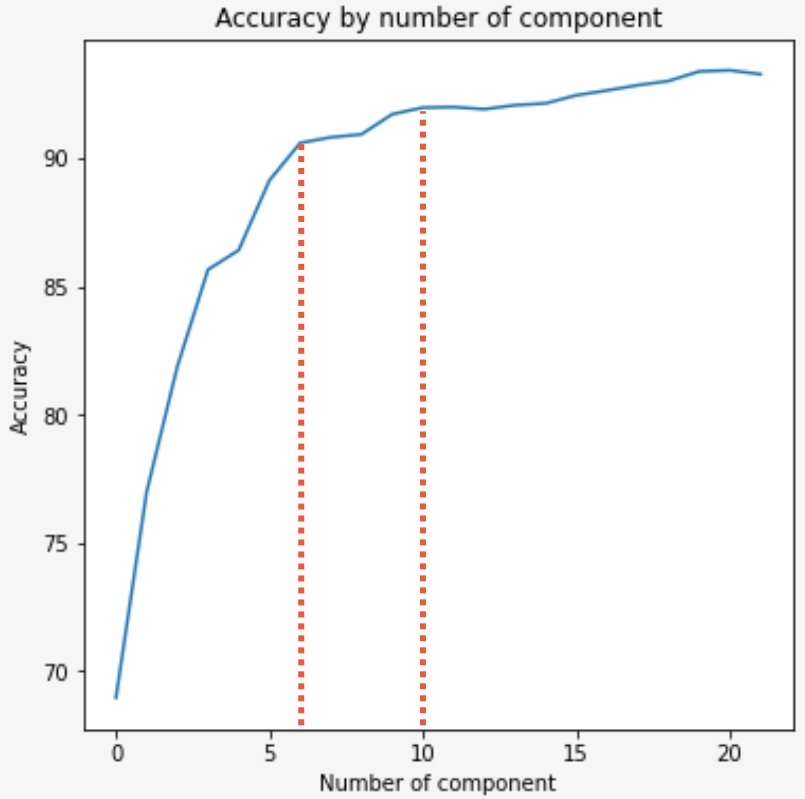
With only 5 components:
- Most of the models could achieve >80% accuracy except the Logistic Regression model.
- Random Forest obtains >90% accuracy

No. of component	Model	Accuracy
5	DecisionTreeClassifier	87.4%
10	DecisionTreeClassifier	88.64%
15	DecisionTreeClassifier	89.19%
20	DecisionTreeClassifier	89.95%
5	GaussianNB	80.61%
10	GaussianNB	81.84%
15	GaussianNB	80.58%
20	GaussianNB	80.35%
5	KNeighborsClassifier	90.12%
10	KNeighborsClassifier	91.42%
15	KNeighborsClassifier	92.04%
20	KNeighborsClassifier	92.63%
5	LogisticRegression	79.06%
10	LogisticRegression	82.62%
15	LogisticRegression	83%
20	LogisticRegression	83.54%
5	RandomForestClassifier	90.86%
10	RandomForestClassifier	92.13%
15	RandomForestClassifier	92.8%
20	RandomForestClassifier	93.45%
5	SVC	88.8%
10	SVC	91.78%
15	SVC	93.63%
20	SVC	94.67%

PCA

We also tried to find out the accuracy by each no. of the component.

```
[ ]  from sklearn.decomposition import PCA

     n_com = []
     for i in range(1,23):
         pca = PCA(n_components = i)
         X_train_pca = pca.fit_transform(X_train_sen)
         X_test_pca = pca.transform(X_test)
         model_rf.fit(X_train_pca, y_train_sen)
         y_pred = model_rf.predict(X_test_pca)
         n_com.append(accuracy_score(y_test, y_pred)*100)
```



The curve shows that 6 and 10 components would have sufficed for getting ~91% and 93% accuracy. Only applying 6 or 10 components is more efficient with limited time or computing resource.

```
[ ]  n_com_dropdown = [68.96827841084078, 76.98490914690484, 81.87249769017555, 85.64829072990453, 86.40899291653834,
                       89.125346473668, 90.58515552817987, 90.80381890976285, 90.92085001539883, 91.70311056359716,
                       91.96181090237143, 91.98336926393594, 91.90021558361565, 92.05420388050509, 92.12811826301201,
                       92.44225438866646, 92.63012011087157, 92.83338466276562, 92.99661225746844, 93.37234370187866,
                       93.4154604250077, 93.26147212811826]
```

HYPER PARAMETERS OPTIMIZATION

Given that the Random Forest performs the best in accuracy, we further try optimizing its hyper parameters to improve the result.

```
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 15)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 16)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 20]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4, 8]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

TOTAL COMBINATIONS OF PARAMETERS = 15 * 16 * 4 * 4 * 2 = 7680

HYPER PARAMETERS OPTIMIZATION

Total combinations of parameters = 15 * 16 * 4 * 4 * 2 = 7680
If finishing one combination takes 10s, the whole process (using Grid Search) would consume ~ 22 hours (assuming the computing resources remain unchanged.

Hence, we tried RandomizedSearch CV, with 25 fits, takes us around an hour to complete the computation eventually. (n_iter = 5; CV = 5)

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,
                               n_iter = 5, cv = 5, verbose=2, random_state=42, n_jobs = -1)
                              # n_iter >> no. of candidates; cv >> no. of folds
# Fit the random search model
rf_random.fit(X_train, y_train)
```

HYPER PARAMETERS OPTIMIZATION

```
1 base_model = RandomForestClassifier(n_estimators = 10, random_state = 42)
2 base_model.fit(X_train, y_train)
3 base_accuracy = evaluate(base_model, X_test, y_test)
```

```
Model Performance
Accuracy = 0.9508
```

```
1 best_random = rf_random.best_estimator_
2 print(best_random)
3 random_accuracy = evaluate(best_random, X_test, y_test)
```

```
RandomForestClassifier(bootstrap=False, max_depth=96, min_samples_leaf=2,
                       n_estimators=1485)
Model Performance
Accuracy = 0.9614
```

```
1 print('Improvement of {:0.2f}%.'.format( 100 * (random_accuracy - base_accuracy) / base_accuracy))
```

```
Improvement of 1.11%.
```

The RandomizedSearchCV returned the best estimator.

With the best parameter combination, the model performance got improved **from 95.08% to 96.14%**, which is a **1.11%** percentage point of improvement.

CONCLUSION

FIRST QUESTION

We can anticipate the customer satisfaction with 96% accuracy by using Random Forest model.

SECOND QUESTION

So far we do not have any factor that is correlated to the satisfaction level, but in the Random Forest model, Inflight Entertainment is the top of feature importance (> 0.200) .

NEXT STEPS

IMPROVE DATA QUALITY

To better anticipate the customer satisfaction level, in real life cases, we would need more features (e.g time, aircraft types, destination coverage).

IMPROVE PREDICTION EFFICIENCY

We could dig deeper to see the increment of runtime when component no. involved increases, so that the runtime/accuracy ratio could be optimized.

Q&A

Thankyou!