25 July 21

Hotel Cancellation Prediction







Project Notes 2 – Machine Learning

Sivaramakrishanan s – july 2020 E

Table of Contents

[Recap of Project Notes 1: 0](#_Toc78143646)

[Study on the problem statement: 0](#_Toc78143647)

[EDA on the Data 0](#_Toc78143648)

[AI and ML on the Hotel Industry 0](#_Toc78143649)

[GOAL 0](#_Toc78143650)

[Approach On The Model Building 1](#_Toc78143651)

[Pre-process the variable for the model building 0](#_Toc78143652)

[One Hot Encoding 0](#_Toc78143653)

[Validate the Relationship 1](#_Toc78143654)

[Identify the Correlation of the Predictors Using (VIF) 2](#_Toc78143655)

[Split Data for Training and Test data 2](#_Toc78143656)

[Scaling 3](#_Toc78143657)

[Target Variable 3](#_Toc78143658)

[Solution 1: Model Building and Interpretation 4](#_Toc78143659)

[Model building 4](#_Toc78143660)

[Compare Model Performance Metrics 4](#_Toc78143661)

[Model Interpretation 6](#_Toc78143662)

[Solution 2: Model Tuning and business implication 6](#_Toc78143663)

[Model Tuning 6](#_Toc78143664)

[Ensemble modelling, wherever applicable 4](#_Toc78143665)

[Compare Model Performance Metrics 0](#_Toc78143666)

[Optimum Model Interpretation 2](#_Toc78143667)

[Implication on the business 2](#_Toc78143668)

Table of Tables

[Table 1 Pre-Processing Binary 0](#_Toc78143596)

[Table 2 Dropped Variables 0](#_Toc78143597)

[Table 3 High Correlation on Target 1](#_Toc78143598)

[Table 4 High VIF Variables 2](#_Toc78143599)

[Table 5 Target Split on Train and Test 3](#_Toc78143600)

[Table 6 Performance Metrics Comparison – BASE Models 4](#_Toc78143601)

[Table 7 Compare Final Tuned Model Performance 0](#_Toc78143602)

Table of Figures

[Figure 1 Correlation on Target Variable 1](#_Toc78143571)

[Figure 2 Correlation on Predators 2](#_Toc78143572)

[Figure 3 Target Distribution 3](#_Toc78143573)

[Figure 4 Performance Metrics - Logistic Regression 4](#_Toc78143574)

[Figure 5 Performance Metrics - LDA 5](#_Toc78143575)

[Figure 6 Performance Metrics - Random Forest 0](#_Toc78143576)

[Figure 7 Performance Metrics - Gaussian NB 0](#_Toc78143577)

[Figure 8 Performance Metrics - KNN 1](#_Toc78143578)

[Figure 9 Performance Metrics - ANN 2](#_Toc78143579)

[Figure 10 Performance Metrics - XGBoost 3](#_Toc78143580)

[Figure 11 AUC Comparison 4](#_Toc78143581)

[Figure 12 Confusion Matrix Comparison 5](#_Toc78143582)

[Figure 13 Interpretation of Test Scores. Base Models 6](#_Toc78143583)

[Figure 14 Performance Metrics RF - Tuned 0](#_Toc78143584)

[Figure 15 Performance Metrics - XGB Tuned 1](#_Toc78143585)

[Figure 16 Performance Metrics - ANN Tuned 2](#_Toc78143586)

[Figure 17 KNN MCE Performance 3](#_Toc78143587)

[Figure 18 Performance Metrics - KNN Tuned 3](#_Toc78143588)

[Figure 19 Performance Metrics - Bagging 4](#_Toc78143589)

[Figure 20 Ada Boosting 5](#_Toc78143590)

[Figure 21 Gradient Boosting 6](#_Toc78143591)

[Figure 22 ROC Comparison 0](#_Toc78143592)

[Figure 23 Confusion Matrix comparison 0](#_Toc78143593)

[Figure 24 Interpretation Of Test Scores - Tuned Models 1](#_Toc78143594)

[Figure 25 Test Implication on the Accuracy 1](#_Toc78143595)

# **Recap of Project Notes 1:**

From the Project notes 1 the major problems of the Hotel Cancellation are detailed analyse has been identified. Below are the high level details about the EDA:

## Study on the problem statement:

* How the transformation of the Hotel booking via OTA(Online Travel Agencies) have impact to the tridtional way of Hotel Booking using the below market research.
  + The industry the free cancellation policy hit 38% and 62% of no-refund policy on hotels where before the pandemic situation(COVID-19).
  + The average cancellation rate decreased from 41.3% to 39.6% (2014 to 2018)
  + The direct financial consequences of cancellations cause the operational problems like over or understaffing.
  + The above problems may lead to decrease customer satisfaction and negative reviews and lead to the booking cancellation.

### EDA on the Data

* The Data provided has 119390 data points with the 27 variables we have done the EDA to analysis the data on the induviual variables and impact in the target variables.
  + Based on the reivewer comments the preprocessing such us outlier / missing value treatment has been acomlished before validating the target variable comparision.
  + The Univarient and Bivarient analysis has been carried out on the analysis.
  + There are new variables created and some of the variables are transformed for the better understanding of the data and the features are not required for analyis are dropped from the dataset. Below are eg.
    - “Lead Time”(New Variable) – Difference of Booking date and Arrival date.
    - “Booking Via Company/Agent”(Variable Transfomation) – To find the wheter the booking is done via company or not / Agent or not.
  + The most importanat variables are further reviewed and identified the Insigts and recommendation.

### AI and ML on the Hotel Industry

We also saw how Artificial intelligence is playing an important role in hospitality management. primarily because of its ability to carry out traditionally human functions at any time of the day.

This potentially means that hotel owners can save significant money, eliminate human error and deliver superior service.

An example of this has been seen with the Dorchester Collection hotel chain, which has made use of the Metis AI platform. By using this technology, the company has been able to sort through data collected via surveys, online reviews etc. and the AI has been able to then analyse this to draw conclusions about overall performance.

### GOAL

Our goal is to build the model to predict the cancellation rate of the hotel type(type1 and type2) using machine learning algorithms on the pre-processed data during EDA.

## **Approach On The Model Building**

1. **Pre-process the variable for the model building**
2. **Review the relationship between predictors and target variables**
3. **Split the data for training and test data set**
4. **Now build the below models using default parameters.(Base Models)**
   * **Logistic Regression**
   * **LDA (LinearDiscriminantAnalysis**)
   * **Navie Bayes**
   * **Random Forest**
   * **KNN – K Nearest Neighbour**
   * **Artificial Neural Networks (ANN)**
   * **XG-Boost**
5. Model tuning using hyper parameters which has a significantly high scores. (We have taken below four models based on the results)
   * **Random Forest**
   * **KNN – K Nearest Neighbour**
   * **Artificial Neural Networks (ANN)**
   * **XG-Boost**
6. Below ensemble methods will be used for model building.
   * Bagging
   * Ada-Boosting
   * Gradient-Boosting
7. Compare the below metrics for the Base, Tuned and Ensemble models
   * Scores (Precision, Recall, F-Accuracy and AUC)
   * Confusion Matrix
   * ROC Curve
8. Interpretation of the most optimum model and its implication on the business

## **Pre-process the variable for the model building**

The variables not as numeric needs to be pre-processed as numeric before scaling and model building. There are difference kinds of encoding technics are used to convert the variables.

Below are the variables are converted as a binary variable (0 or 1) based on the value.

Table 1 Pre-Processing Binary

|  |  |  |
| --- | --- | --- |
| Features | Value to 1 | Value to 0 |
| hotel | Type1 | Type2 |
| previous\_cancellations\_new | Yes | No |
| previous\_bookings\_not\_canceled\_new | Yes | No |
| previous\_cancellations\_encoded | Booking Via Company | Booking Not Via Company |
| book\_by\_agent | Agent Booking | Not Agent Booking |

### **One Hot Encoding**

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

Below categorical variables are one hot encoding.

* Deposit Type
* Market Segment
* Distribution Channel
* Customer Type
* Assigned Room Type
* Location
* Meal

The “booking date” and “arrival date” are converted using the string replace function.

**Example :** 2017-07-24 converted as **20170724**

The below variables are dropped as converted as part of pre-processing and few variables are not required for ML.

Table 2 Dropped Variables

|  |  |  |
| --- | --- | --- |
| Deposit Type  Market Segment  Distribution Channel  Customer Type  Assigned Room Type  Location  Hotel  Previous Cancellations New  Previous Bookings Not Canceled New  Book Via Company  Book By Agent  Company | Arrival Date  Booking Changes  Book By Agent Encoded  Country Type  C Name  Country  Leadtime Month  Previous Cancellations  Is Repeated Customer  Previous Bookings Not Canceled  Children  Stays In Weekend Nights | Meal  Total Guest  Days In Waiting List  Adults  A Day Of Week  A Month  A Day  A Weekno  A Year  Reserved Room Type  Total Stays  Stays In Week Nights |

### **Validate the Relationship**

Visualizing correlation coefficients between features and cancellation:

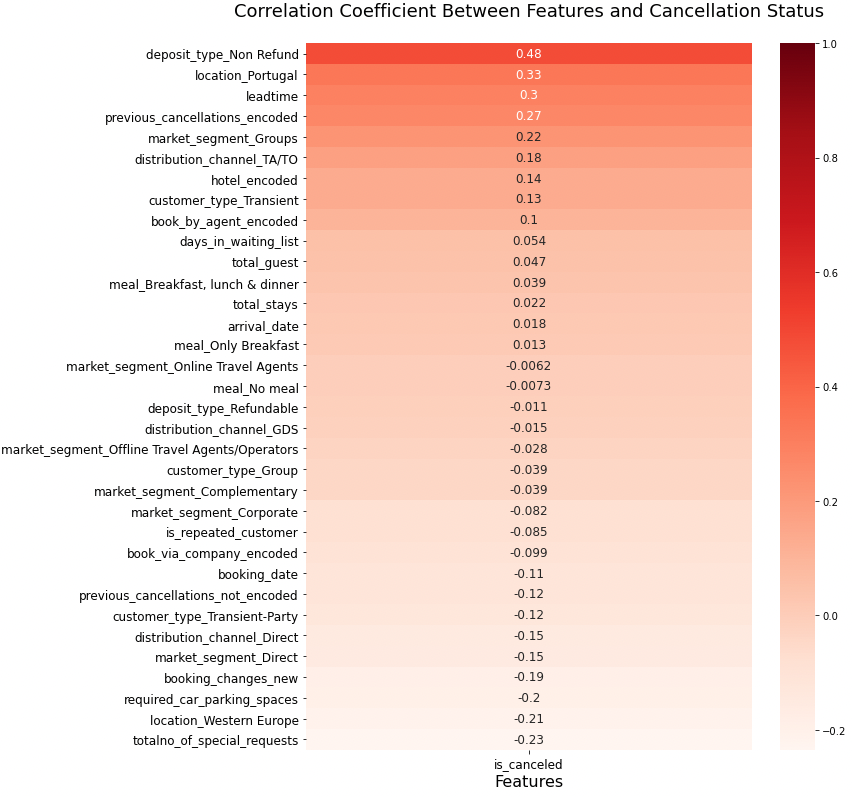


Figure 1 Correlation on Target Variable

The below variables are have optimum correlation on the target variable. We have taken the threshold as 0.2 to generate the below table.

Table 3 High Correlation on Target

|  |  |
| --- | --- |
| Features | Corr(is\_canceled) |
| deposit\_type\_Non Refund | 0.481457 |
| location\_Portugal | 0.331595 |
| leadtime | 0.295173 |
| previous\_cancellations\_encoded | 0.270943 |
| market\_segment\_Groups | 0.221859 |
| location\_Western Europe | -0.212401 |
| totalno\_of\_special\_requests | -0.234658 |

### Identify the Correlation of the Predictors Using (VIF)

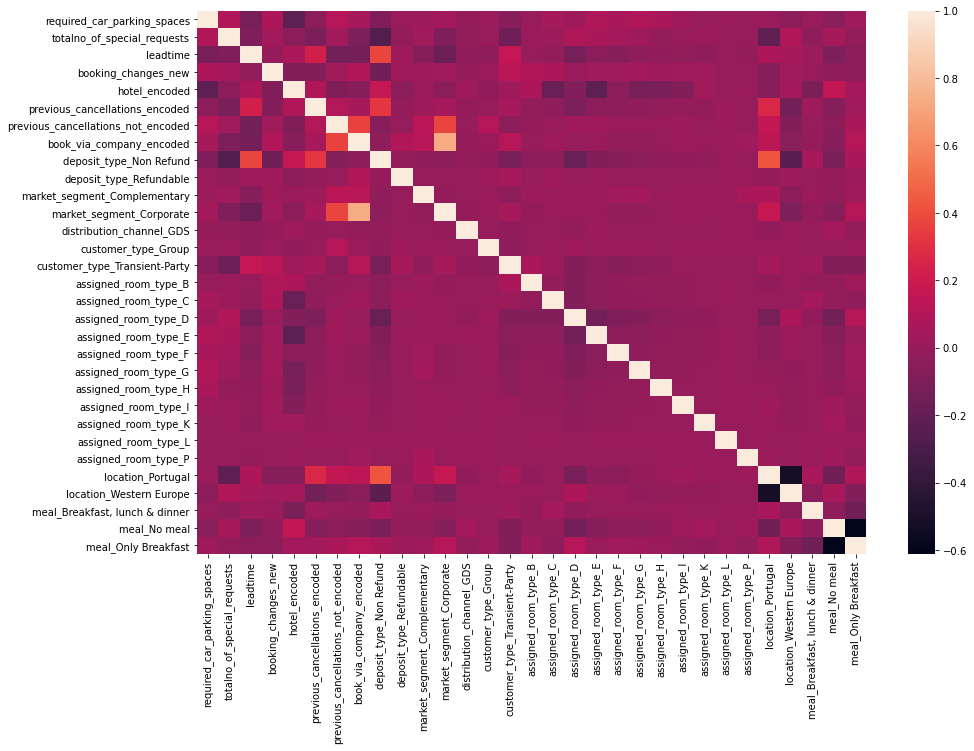


Figure 2 Correlation on Predators

From the heatmap there is no multi correction in the majority of variables. Below are variables have high correction between predictors this will kept for model building. We have tried

Table 4 High VIF Variables

|  |  |
| --- | --- |
| Features | Multi-correlation |
| market\_segment\_Direct | 16.36 |
| market\_segment\_Groups | 18.62 |
| market\_segment\_Offline Travel Agents/Operators | 22.47 |
| market\_segment\_Online Travel Agents | 54.63 |
| distribution\_channel\_Direct | 12.41 |
| distribution\_channel\_TA/TO | 56.46 |
| customer\_type\_Transient | 23.65 |

## Split Data for Training and Test data

To build the machine learning models required to split the dataset into Training and Testing Data with the ratio of 85:15. Then the sliced datasets are stored in two variables as X\_train and X\_test. The Random State “9” is used for split the data.

The target variable “is\_canceled” is dropped in X dataset and stored in y dataset for the verification of the model performance.

Table 5 Target Split on Train and Test

|  |  |  |
| --- | --- | --- |
| **Class** | **Train** | **Test** |
| Not Cancelled (0) | 63891 | 11275 |
| Cancelled (1) | 37590 | 6634 |

The X\_train has **101481** values and X\_test as **17909** and below is the target distribution.

## Scaling

Standardization or scaling is an important aspect of data pre-processing, it is applied to independent variables which helps to normalise the data in a particular range. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

**Feature Scaling** is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

The [Machine Learning algorithms](https://intellipaat.com/blog/tutorial/machine-learning-tutorial/machine-learning-algorithms/) that require the feature scaling are mostly KNN (K-Nearest Neighbours), Neural Networks, Linear Regression, and Logistic Regression.

The machine learning algorithms that do not require feature scaling is mostly non-linear ML algorithms such as Decision trees, Random Forest, AdaBoost, Naïve Bayes, etc.

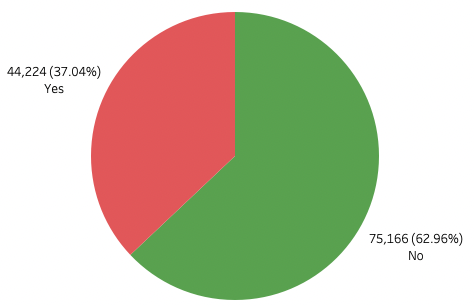
This data is almost balance, so later on for the machine learning process we won’t need to do an imbalance handling.

## Target Variable

Let’s begin with the Target variable distribution on the Hotel Cancellation status. This data represented in the feature is\_canceled.

Cancelled : The count of Cancelled distribution is 44224 which is 37.04% of the total data.

Not Cancelled : The count of Not Cancelled distribution is 75166 which is 62.96% of the total data.



* **The rate of cancellation is likely matching with the industry standard which is around 37% - 40% (Source:**[**Emerchantpay Link**](https://www.emerchantpay.com/infographic-how-can-hotels-combat-rising-cancellation-rates/)**)**
* **The problem that hospitality industries are facing that there are almost 4 cancellation in every 10 bookings**
* **The target data is almost balance, so later on for the machine learning process we won’t need to do an imbalance handing**

Figure 3 Target Distribution

## 

## Solution 1: Model Building and Interpretation

### Model building

Question : Build various models (You can choose to build models for either or all of descriptive, predictive or prescriptive purposes)

#### Logistic Regression – Base Model

Logistic regression is a special case of linear regression where we only predict the outcome in a categorical variable. It predicts the probability of the event using the log function. The equation:  **p = 1 / 1 + e-(β0 + β1X1 + β2X2 …. + βnXn)** variable. It predicts the probability of the event using the log function. This is a not majorly impacted by scaling but in our case, we are going to use the scaled data for the prediction.

The model building of the Logistic regression used the Random State as 0 and the below default parameters.

##### Default Parameters:

penalty='l2', dual=False, tol=0.0001, C=1, fit\_intercept=True, intercept\_scaling=1, class\_weight=None, solver='lbfgs', max\_iter=100, multi\_class='auto', verbose=0, warm\_start=False, n\_jobs=None, l1\_ratio=None

##### Performance Metrics

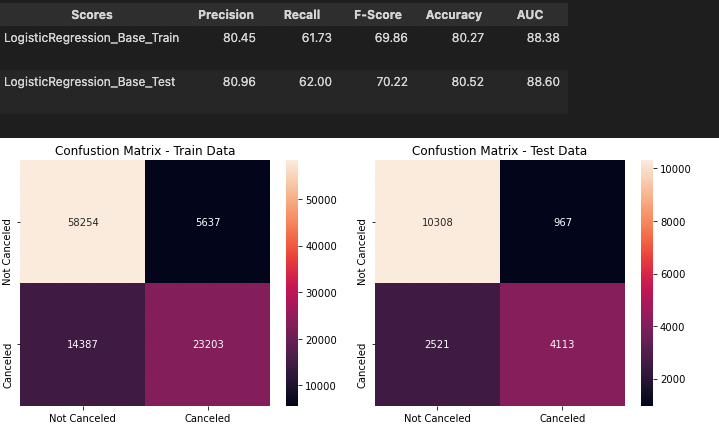


Figure 4 Performance Metrics - Logistic Regression

##### Inferences:

* This model is not overfit or underfit (the training and testing scores are close together)
* The model is outperforming the baseline with a **testing accuracy of 80.52%**

#### LDR (Linear Discriminant Analysis) – Base Model

**Linear Discriminant Analysis** or **Normal Discriminant Analysis** or **Discriminant Function Analysis** is a dimensionality reduction technique which is commonly used for the supervised classification problems.

Under LDA we assume that the density for X, given every class k is following a Gaussian distribution. Here is the density formula for a multivariate Gaussian distribution:

p is the dimension and  is the covariance matrix. This involves the square root of the determinant of this matrix. In this case, we are doing matrix multiplication. The vector x and the mean vector μk are both column vectors.

For Linear discriminant analysis (LDA): .

The model building has built using below default parameters and Random State as 0.

##### Default Parameters:

solver='svd', shrinkage=None, priors=None, n\_components=None, store\_covariance=False, tol=0.0001, covariance\_estimator=None

##### Performance Metrics

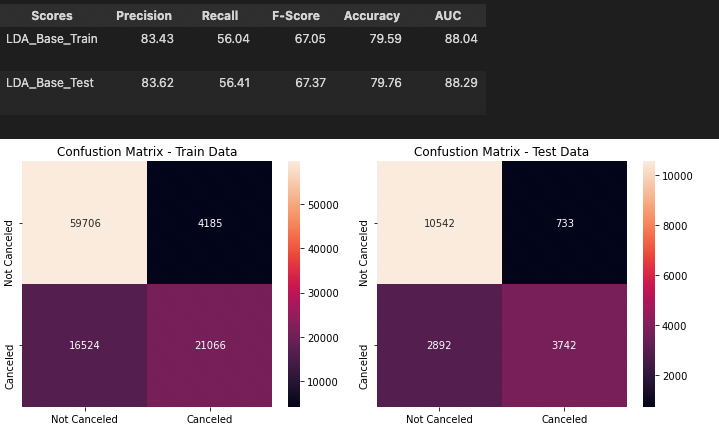


Figure 5 Performance Metrics - LDA

##### Inferences:

* This model is same as Logistic regression and not overfit or underfit (the training and testing scores are close together)
* The model is outperforming the baseline with a **testing accuracy of 79.76%** which is less then Logistic Regression

#### Random Forest Classifier – Base Model

Random Forest is another Supervised Learning Technique used in Machine Learning which consists of many decision trees that helps in predictions using individual trees and selects the best output from them.

Using the Train Dataset(**X\_train**) we will be creating a Random Forest model and then further testing the model on Test Dataset(**X\_test**)

For creating the Random Forest, the package “**RandomForestClassifier**” is imported from **sklearn.metrics**.

The model building has built using below default parameters and Random State as 0.

##### Default Parameters:

n\_estimators=100, criterion="gini", max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0, max\_features="auto", max\_leaf\_nodes=None, min\_impurity\_decrease=0, min\_impurity\_split=None, bootstrap=True, oob\_score=False, n\_jobs=None, verbose=0, warm\_start=False, class\_weight=None, ccp\_alpha=0, max\_samples=None

##### Performance Metrics



Figure 6 Performance Metrics - Random Forest

##### Inferences:

* This model is a overfit model and the training score is greater than testing scores.
* The model is performing the baseline with a **testing accuracy of 88.61%.**

#### Naïve Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable. Bayes’ theorem states the following relationship, given class variable *y* and dependent feature vector *x1* through *xn*, :

[**GaussianNB**](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB) implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be

**Gaussian:**

Navies Bayes not affected on the scaling so the original data used for the model building.

The model building has built using below default parameters and Random State as 0.

##### Default Parameters:

priors=None, var\_smoothing=1e-9

##### Performance Metrics

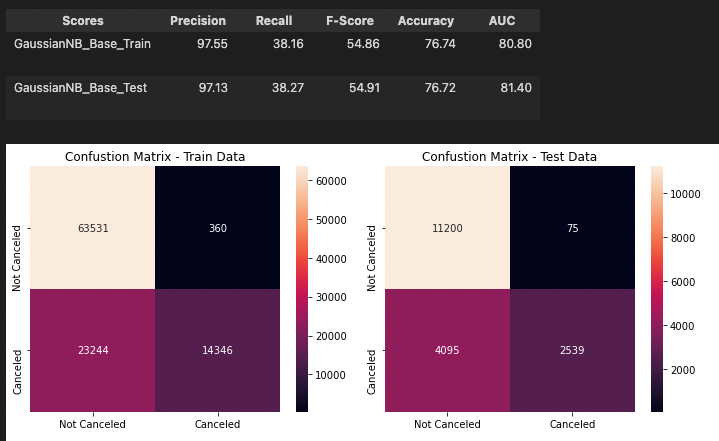


Figure 7 Performance Metrics - Gaussian NB

##### Inferences:

* This model is same not overfit or underfit (the training and testing scores are close together)
* The model is outperforming the baseline with a **testing accuracy of 76.72%** which is less then Logistic Regression & LDA

#### KNN – Base Model

KNN is a lazy learning, non-parametric algorithm. It uses data with several classes to predict the classification of the new sample point. KNN is non-parametric since it doesn’t make any assumptions on the data being studied, i.e., the model is distributed from the data.

KNN is a lazy model due to below reasons.

**KNeighborsClassifier** function to be imported from the **sklearn.neighbors** module to proceed to build the model. We need to choose the nearest data points (the value of K). K(**n\_neighbors**) can be any integer.

For the base model we are going to use the K value as (**n\_neighbors=3)** to train the model.

**KNN** algorithm is seriously affected on unscaled data because you choose the KK closest samples for your predictions. If one of the features has large values (e.g. ≈≈ 1000), and the other has small values (e.g. ≈1≈1), your predictions will favor the feature with large values because the distance calculated will be dominated with it. We are going to use the scaled data(**X\_train\_sc**) to train the model.

The model building has built using below default parameters.

##### Default Parameters:

n\_neighbors=5, weights='uniform', algorithm='auto', leaf\_size=30, p=2, metric='minkowski', metric\_params=None, n\_jobs=None,

##### Performance Metrics

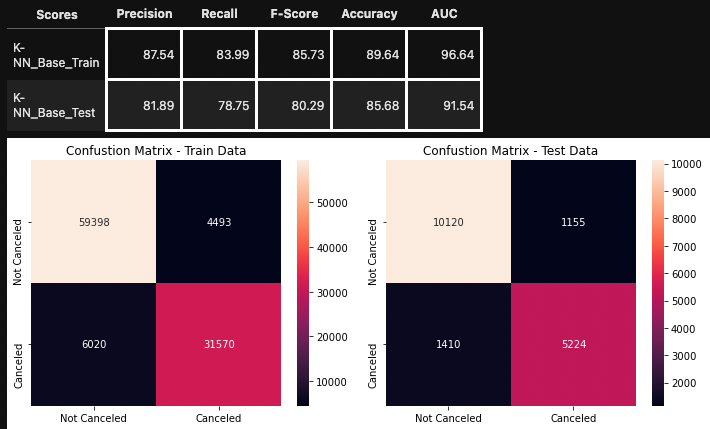


Figure 8 Performance Metrics - KNN

##### Inferences:

* This model is same as medium variance on accuracy which is good to proceed with model. The training and testing scores are not closer or not much far.
* The model is outperforming the baseline with a **testing accuracy of 85.68%** which is less then Logistic Regression

#### Artificial Neural Networks (ANN)

An artificial neural network (ANN) is the component of artificial intelligence that is meant to simulate the functioning of a human brain. Processing units make up ANNs, which in turn consist of inputs and outputs. The inputs are what the ANN learns from to produce the desired output. Backpropagation is the set of learning rules used to guide artificial neural networks.

The practical applications for ANNs are far and wide, encompassing finance, personal communication, industry, education.

The Train dataset(**X\_train**) and test dataset(**X\_test**) will be used to creating a Neural Network using **MLPClassifier** from **sklearn.metrics**.

##### Default Parameters:

hidden\_layer\_sizes=(100, ), activation="relu", \*, solver='adam', alpha=0.0001, batch\_size='auto', learning\_rate="constant", learning\_rate\_init=0.001, power\_t=0.5, max\_iter=200, shuffle=True, random\_state=None, tol=0.0001, verbose=False, warm\_start=False, momentum=0.9, nesterovs\_momentum=True, early\_stopping=False, validation\_fraction=0.1, beta\_1=0.9, beta\_2=0.999, epsilon=1e-8, n\_iter\_no\_change=10, max\_fun=15000

##### Performance Metrics

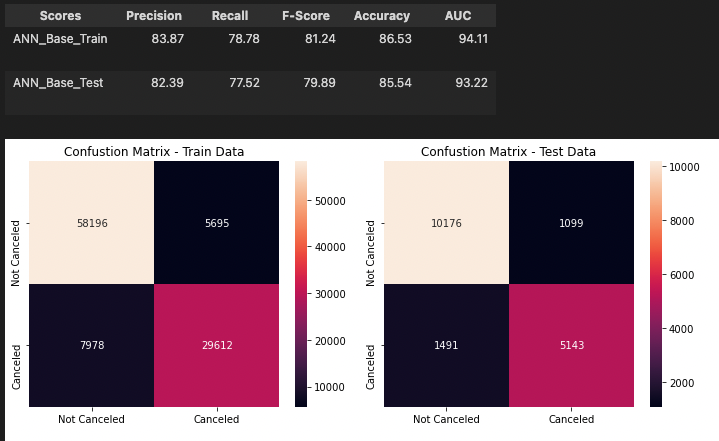


Figure 9 Performance Metrics - ANN

##### Inferences:

* This model is same not overfit or underfit (the training and testing scores are close together)
* The model is outperforming the baseline with a testing accuracy of 85.54%
* So far this the better fit model from the previous models what we compared.

#### XGBoost

XGBoost or extreme gradient boosting is one of the well-known [gradient boosting](https://analyticsindiamag.com/gradient-descent-everything-you-need-to-know-with-implementation-in-python/)techniques(ensemble) having enhanced performance and speed in tree-based (sequential decision trees) machine learning algorithms.

The Train dataset(**X\_train**) and test dataset(**X\_test**) will be used to creating a XGBoost using **XGBoostClassifier** from **xgboost.**

##### Default Parameters:

objective="binary:logistic", use\_label\_encoder=True

##### Performance Metrics

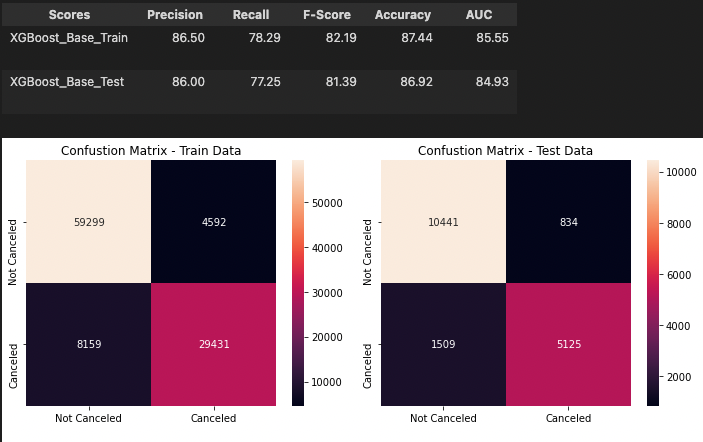


Figure 10 Performance Metrics - XGBoost

##### Inferences:

* This model is same not overfit or underfit (the training and testing scores are close together)
* The model is outperforming the baseline with a testing accuracy of 86.92%
* This model has lowest variance with compare to all the models up on all base models.

### Compare Model Performance Metrics

Question:. Test your predictive model against the test set using various appropriate performance metrics

#### Identifying the Metrics

Accuracy is one of the common evaluation metrics in classification problems, that is the total number of correct predictions divided by the total number of predictions made for a dataset. Accuracy is useful when the target class is *well balanced* but is not a good choice with unbalanced classes.

From the initial analysis there is no class imbalance (37% : 63%) so the standard approach of the accuracy has been taken as metric for this problem.

**Other approach:** During analysis we see that there are duplicated values nearly 33k which is 27.82% of total data. After removing the duplicated data there is a high class imbalance and the model metrics are reduced. In this case can take Recall or F1-Score as metric and using AUC and ROC threshold improve the metrics as Recall or F1-Score by decrees the FP values.

Table 6 Performance Metrics Comparison – BASE Models

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **RandomForest** | | **K-NN** | | **XGBoost** | | **ANN** | | **GaussianNB** | | **LogisticRegression** | | | **LDA** | | |
|  | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | | **Test** |
| **Precision** | 99.36 | 86.05 | 87.54 | 81.89 | 86.50 | 86.00 | 83.87 | 82.39 | 97.55 | 97.13 | 80.45 | 80.96 | 83.43 | | 83.62 |
| **Recall** | 98.83 | 82.65 | 83.99 | 78.75 | 78.29 | 77.25 | 78.78 | 77.52 | 38.16 | 38.27 | 61.73 | 62.00 | 56.04 | | 56.41 |
| **F-Score** | 99.09 | 84.31 | 85.73 | 80.29 | 82.19 | 81.39 | 81.24 | 79.89 | 54.86 | 54.91 | 69.86 | 70.22 | 67.05 | | 67.37 |
| **Accuracy** | 99.33 | 88.61 | 89.64 | 85.68 | 87.44 | 86.92 | 86.53 | 85.54 | 76.74 | 76.72 | 80.27 | 80.52 | 79.59 | | 79.76 |
| **AUC** | 99.94 | 95.42 | 96.64 | 91.54 | 85.55 | 84.93 | 94.11 | 93.22 | 80.80 | 81.40 | 88.38 | 88.60 | 88.04 | | 88.29 |

|  |  |
| --- | --- |
|  | Figure 11 AUC Comparison |

**

Figure 12 Confusion Matrix Comparison

### Model Interpretation

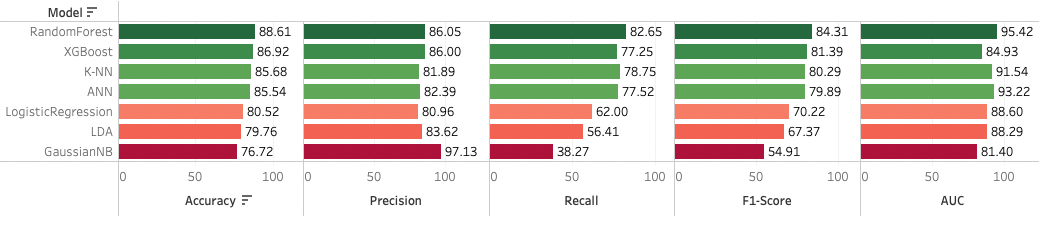
Question : Interpretation of the model(s)

Figure 13 Interpretation of Test Scores. Base Models

* The models Random Forest, KNN and ANN models are overfit upon ANN having a minimum variance. Overall these models are given optimum performance values so we consider these models for the model tuning.
* The GaussianNB, LR and LDR have minimum variance where NB model have a high precision which is unable to predict which is actually cancelled which causes the higher Type 2 error.
* XGBoost is one of the best fit model on over all models where variance in accuracy is moderate.
* Random Forest has a highest Accuracy upon all the models and followed by XGBoost, KNN and ANN.
* Considering the interpretations we consider the models Random Forest, KNN, ANN and XGBoost for the model tuning since the other models are not provided the significant accuracy.

## Solution 2: Model Tuning and business implication

Question : Any other model tuning measures(if applicable)

### **Model Tuning**

Tuning is the process of maximizing a model’s performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate “hyperparameters.”

Hyperparameters can be classified as model hyperparameters, that cannot be inferred while fitting the machine to the training set because they refer to the model selection task, or algorithm hyperparameters.

Using the **GridSearchCV** package from **sklearn.model\_selection** we will identify the best parameters to build the below models.

* + **Random Forest**
  + **KNN – K Nearest Neighbour**
  + **Artificial Neural Networks (ANN)**
  + **XG-Boost**

Hence, doing a few iterations with the values we got the best parameters to build the best models which available below.

#### Random Forest Classifier – Tuned Model

On high level, based on the base model metrics the model has issue with overfitting since we have reduced the n\_estimater from default value (100) also limit the max number of features to 15 or 16.

Created a user defined function as “rfrun” with the possible parameters can be passed and tune the model. Let’s apply below hyper parameters and try to build the Tuned model and check how its impacted the results.

|  |  |
| --- | --- |
| Hyper Parameters: | Best Estimators: |
| * max\_depth' : [35,48] * 'max\_features' : [15,16] * 'min\_samples\_leaf' : 1, * 'min\_samples\_split' : 5 * 'n\_estimators' : [75,85] | * max\_depth' : 35 * 'max\_features' : 16 * 'min\_samples\_leaf' : 1 * 'min\_samples\_split' : 5 * 'n\_estimators' : 75 |

Above are best estimators are best estimators identified by the GridSearchCV function. The process took nearly 04:00 mins to identify the best parameters from GridSearchCV.

Let’s fit the best estimates on the train and test set and evaluate the model metrics.

##### Performance Metrics

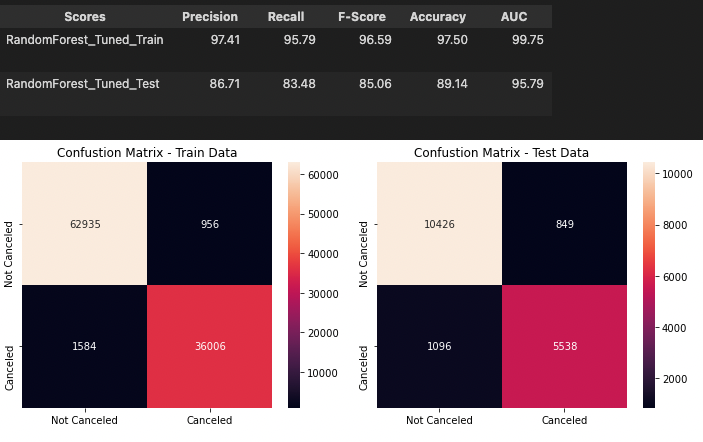


Figure 14 Performance Metrics RF - Tuned

##### Inferences:

* The model is overfit but the variance has been reduced.
* The model is outperforming the baseline with a testing accuracy of 89.14% here the accuracy has increased.
* This is the model have highest accuracy upon all the models.

#### XGBoost Classifier – Tuned Model

The XGBoost base model provided good accuracy as “86.92” on the test and this is with the 10% of the metrics threshold on the train data accuracy “87.44”. Here we are going to apply the below hyper parameters and see how this is impacted the result metrics.

##### Hyper Parameters:

* 'n\_estimators' : [150], # 200, 250],
* 'colsample\_bytree' : [0.7],# 0.8],
* 'max\_depth' : [15],#20,25],
* 'reg\_alpha' : [1.1], #, 1.2, 1.3],
* 'reg\_lambda' : [1.1], #, 1.2, 1.3],
* 'subsample' : [0.7], #, 0.8, 0.9

##### Best Estimators:

XGBClassifier (base\_score=0.5, booster='gbtree', colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=0.7, gamma=0, gpu\_id=-1, importance\_type='gain', interaction\_constraints='', learning\_rate=0.300000012, max\_delta\_step=0, max\_depth=15, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=150, n\_jobs=8, num\_parallel\_tree=1, objective='binary:hinge', random\_state=0, reg\_alpha=1.1, reg\_lambda=1.1, scale\_pos\_weight=None, subsample=0.7, tree\_method='exact', validate\_parameters=1, verbosity=None)

Above are best estimators are best estimators identified by the GridSearchCV function. The process took nearly 00:46 secs to identify the best parameters from GridSearchCV. Let’s fit the best estimates on the train and test set and evaluate the model metrics.

##### Performance Metrics

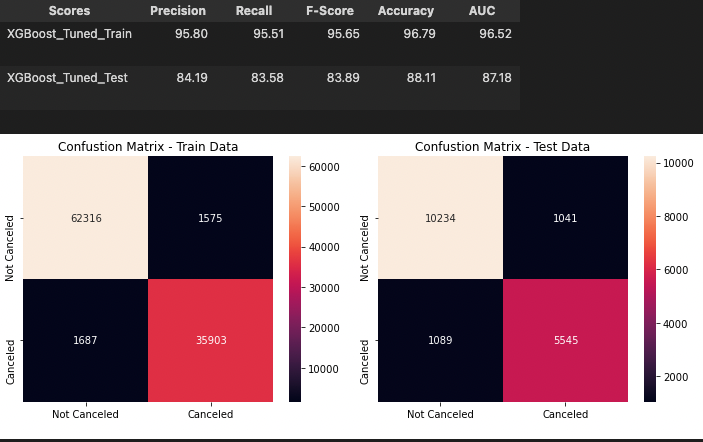


Figure 15 Performance Metrics - XGB Tuned

##### Inferences:

* The model is overfit then base model but the variance has been reduced.
* The model is outperforming the baseline with a testing accuracy of 89.14% here the accuracy has increased.

#### ANN MLPClassifier – Tuned Model

The ANN(MLPClassfier) base model provided good accuracy as “85.53” on the test and this is with the 10% of the metrics threshold on the train data accuracy “85.54”. Here we are going to apply the below hyper parameters and see how this is impacted the result metrics.

|  |  |
| --- | --- |
| Hyper Parameters: | Best Estimators: |
| * 'hidden\_layer\_sizes' : [200,350], * 'max\_iter' : [500,750], * 'solver' : ['sgd','adam'], * 'tol' : [0.01,0.001], | * hidden\_layer\_sizes : 350, * max\_iter : 500, * random\_state : 9, * tol : 0.001 |

Above are best estimators are best estimators identified by the GridSearchCV function. The process took nearly 18:00 mins to identify the best parameters from GridSearchCV.

Let’s fit the best estimates on the train and test set and evaluate the model metrics.

##### Performance Metrics

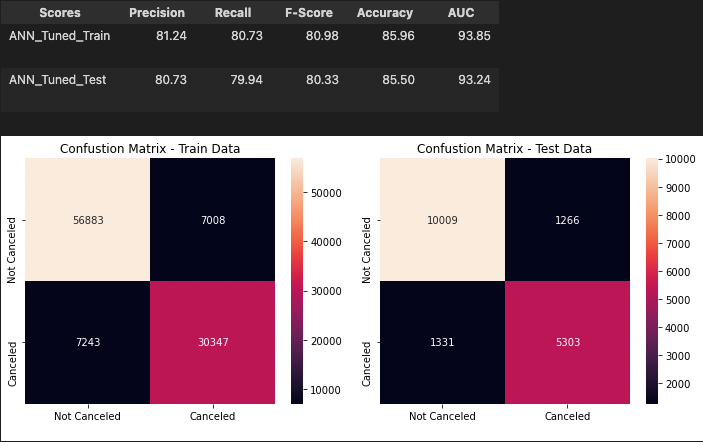


Figure 16 Performance Metrics - ANN Tuned

##### Inferences:

* There is no significant change in the this model on the accuracy after tuning.
* Overall the accuracy has been reduced a bit from 85.54% to 85.50% on the test data.

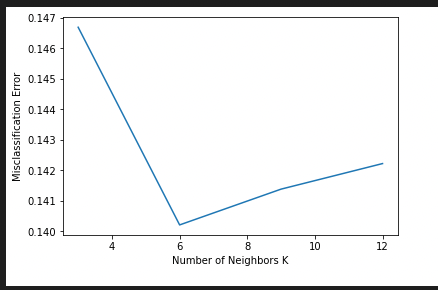
#### KNN KNeighborsClassifier – Tuned Model

The KNN(KNeighborsClassifier) base model provided good accuracy as “85.33” on the test and this is with the 10% of the metrics threshold on the train data accuracy “91.89”. Here we are going to apply the below hyper parameters and see how this is impacted the result metrics.

##### Hyper Parameters:

Since KNN is works based on the n\_neighbors parameter we have built model based on the default parameter as 5 and here we have made the iteration from 3 to 15 with increment value as 3 , So there are 4 different models built on the iteration. Then the MCE (Minimum Classification Error) calculated by subtracting the model accuracy with the value 1 (100%) and the MCE is less that is the good model.

##### Best Estimators:

Based on the models built the best model is with minimum MCE which is KNN = 6 with MCE as “14%”

The process took nearly 04:00 mins to identify the best parameters iterations.

Figure 17 KNN MCE Performance

Let’s fit the best estimates on the train and test set and evaluate the model metrics.

##### Performance Metrics

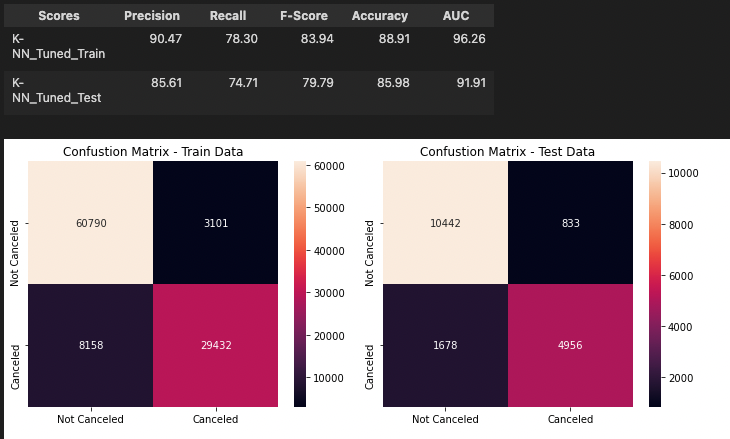


Figure 18 Performance Metrics - KNN Tuned

##### Inferences:

* There is no significant change in the this model on the accuracy after tuning.
* Overall the accuracy has been increased a bit from 85.68% to 85.98% on the test data.

### Ensemble modelling, wherever applicable

#### Bagging Classifier

Bagging, a Parallel ensemble method (stands for Bootstrap Aggregating), is a way to decrease the variance of the prediction model by generating additional data in the training stage. This is produced by random sampling with replacement from the original set. By sampling with replacement, some observations may be repeated in each new training data set. In the case of Bagging, every element has the same probability to appear in a new dataset. By increasing the size of the training set, the model’s predictive force can’t be improved. It decreases the variance and narrowly tunes the prediction to an expected outcome.

The base estimator to fit on random subsets of the dataset. If None, then the base estimator is a [DecisionTreeClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html" \l "sklearn.tree.DecisionTreeClassifier" \o "sklearn.tree.DecisionTreeClassifier). ***We are going use the RF model for Bagging***

##### Parameters Used:

* base\_estimator : rfcl (This is the tuned RF model),
* n\_estimator : 50,
* random\_state : 0,

##### Performance Metrics

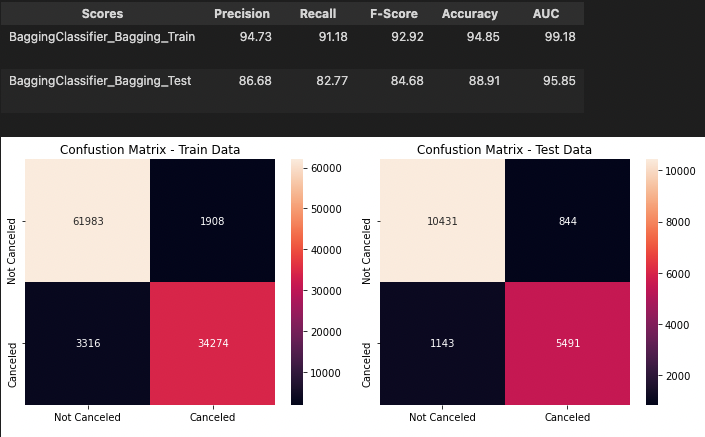


Figure 19 Performance Metrics - Bagging

#### Boosting

Boosting is a sequential ensemble method that in general decreases the bias error and builds strong predictive models. The term ‘Boosting’ refers to a family of algorithms which converts a weak learner to a strong learner.

Boosting gets multiple learners. The data samples are weighted and therefore, some of them may take part in the new sets more often. In each iteration, data points that are mis-predicted are identified and their weights are increased so that the next learner pays extra attention to get them right.

There are several types of Boosting available we are going to use the two types of boosting algorithm.

1. AdaBoosting
2. GradientBoosting

#### 

#### Ada Boosting

**AdaBoost:** AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm that works on the principle of Boosting.

##### Parameters Used:

* n\_estimator : 10,
* random\_state : 0,

##### Performance Metrics

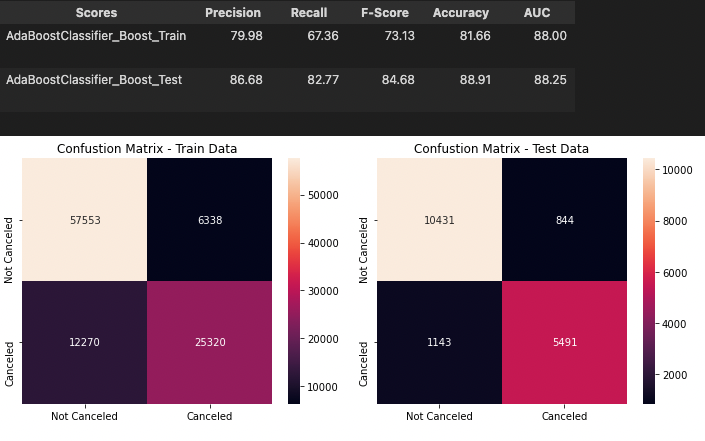


Figure 20 Ada Boosting

#### 

#### Gradient Boosting

Gradient boosting re-defines boosting as a numerical optimisation problem where the objective is to minimise the loss function of the model by adding weak learners using gradient descent.

As gradient boosting is based on minimising a loss function, different types of loss functions can be used resulting in a flexible technique that can be applied to regression, multi-class classification, etc.

##### Parameters Used:

* random\_state : 0,

##### Performance Metrics

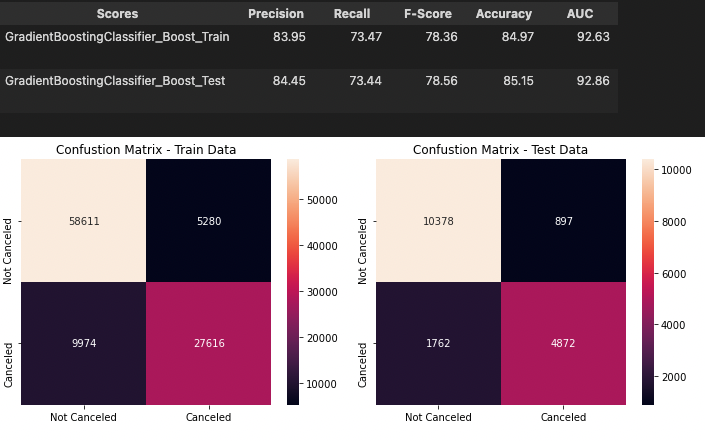


Figure 21 Gradient Boosting

### Compare Model Performance Metrics

Table 7 Compare Final Tuned Model Performance

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **RandomForest** | | | | **XGBoost** | | | | **K-NN** | | | | **ANN** | | | | **Bagging** | | **Boosting** | | | |
|  | **Base** | | **Tuned** | | **Base** | | **Tuned** | | **Base** | | **Tuned** | | **Base** | | **Tuned** | | **Bagging** | | **Gradient** | | **Ada** | |
|  | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** | **Train** | **Test** |
| Precision | 99.36 | 86.05 | 97.41 | 86.71 | 86.50 | 86.00 | 95.80 | 84.19 | 87.54 | 81.89 | 90.47 | 85.61 | 83.87 | 82.39 | 81.24 | 80.73 | 94.73 | 86.68 | 83.95 | 84.45 | 79.98 | 86.68 |
| Recall | 98.83 | 82.65 | 95.79 | 83.48 | 78.29 | 77.25 | 95.51 | 83.58 | 83.99 | 78.75 | 78.30 | 74.71 | 78.78 | 77.52 | 80.73 | 79.94 | 91.18 | 82.77 | 73.47 | 73.44 | 67.36 | 82.77 |
| F-Score | 99.09 | 84.31 | 96.59 | 85.06 | 82.19 | 81.39 | 95.65 | 83.89 | 85.73 | 80.29 | 83.94 | 79.79 | 81.24 | 79.89 | 80.98 | 80.33 | 92.92 | 84.68 | 78.36 | 78.56 | 73.13 | 84.68 |
| Accuracy | 99.33 | 88.61 | 97.50 | 89.14 | 87.44 | 86.92 | 96.79 | 88.11 | 89.64 | 85.68 | 88.91 | 85.98 | 86.53 | 85.54 | 85.96 | 85.50 | 94.85 | 88.91 | 84.97 | 85.15 | 81.66 | 88.91 |
| AUC | 99.94 | 95.42 | 99.75 | 95.79 | 85.55 | 84.93 | 96.52 | 87.18 | 96.64 | 91.54 | 96.26 | 91.91 | 94.11 | 93.22 | 93.85 | 93.24 | 99.18 | 95.85 | 92.63 | 92.86 | 88.00 | 88.25 |

|  |  |
| --- | --- |
|  | Figure 22 ROC Comparison |

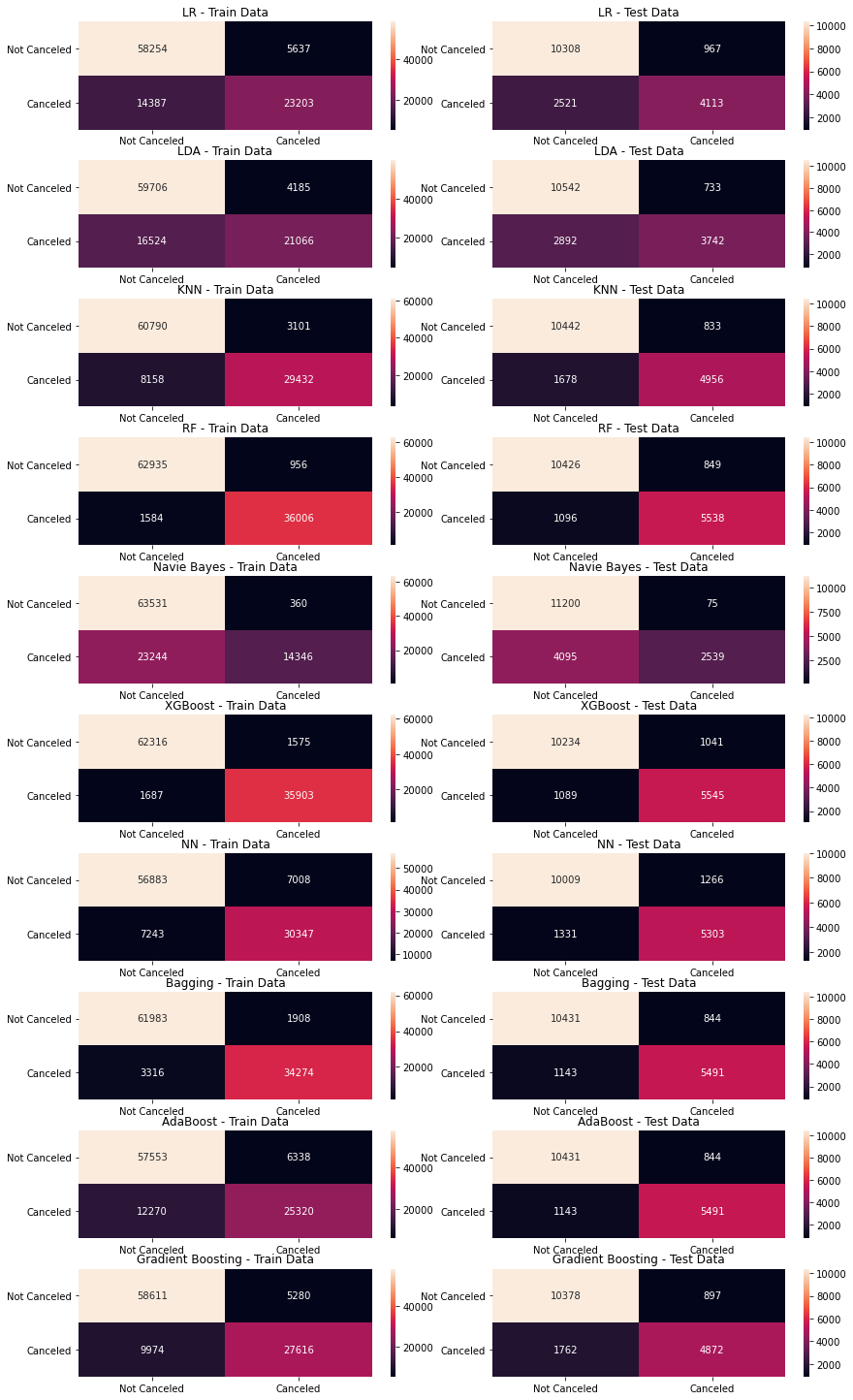


Figure 23 Confusion Matrix comparison

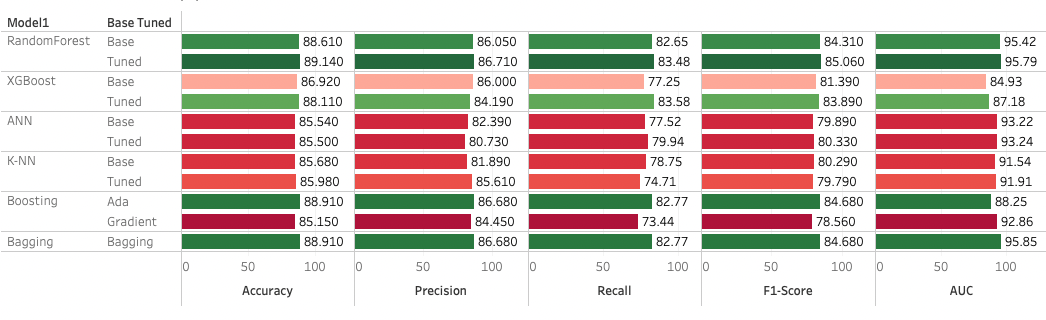


Figure 24 Interpretation Of Test Scores - Tuned Models

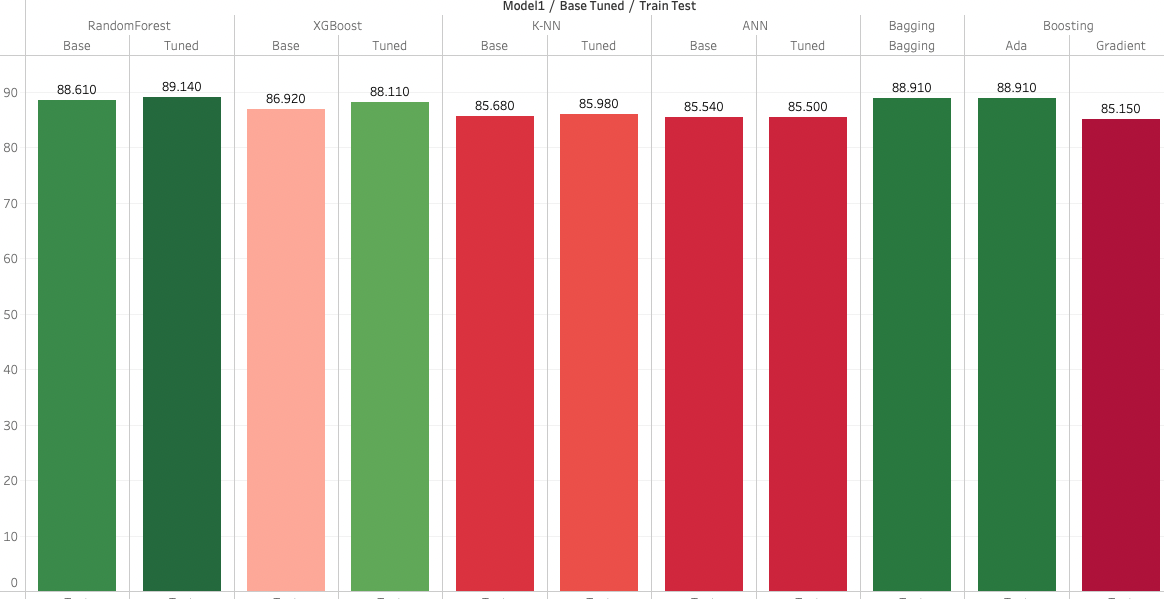


Figure 25 Test Implication on the Accuracy

* Overall the tuned model accuracy has increased also the variance has been reduced.
* Random Forest:
  + The model is overfit but the variance has been reduced.
  + The model is outperforming the baseline with a testing accuracy of 89.14% here the accuracy has increased. This is the model have highest accuracy upon all the models.
* XGB:
  + The model is overfit then base model but the variance has been reduced.
  + The model is outperforming the baseline with a testing accuracy of 89.14% here the accuracy has increased.
* ANN:
  + There is no significant change in the this model on the accuracy after tuning.
  + Overall the accuracy has been reduced a bit from 85.54% to 85.50% on the test data.
* KNN:
  + There is no significant change in the this model on the accuracy after tuning.
  + Overall the accuracy has been increased a bit from 85.68% to 85.98% on the test data.
* Bagging:
  + This Model has good accuracy value and good fit model also the variance is significantly moderate.
* Boosting:
  + Upon both boosting technique the Ada boosting technique has good accuracy value, but this model has a under fitting as test accuracy are higher than train accuracy
  + Gradient boosting is not performed as expected.

Question: Interpretation of the most optimum model and its implication on the business

### Optimum Model Interpretation

* Using Accuracy As The Primary Evaluation Metrics
  + The First Reason Why Accuracy is used as the evaluation metric here because we have somehow a balance data 63 % Confirmed Booking and 37% Canceled Booking in this case our dataset is balance and hence using accuracy is acceptable in this case
  + in this case every class is equally important
* Overfitting
  + For the base model we see that other than Random Forest algorithms that doesn't have an overfitting condition
  + After Hyperparameter Tuning on the XGBoost and Random Forest have overfitting but the overfitting value is the significant difference which is less (10%), and after hyperparameter tuning (Random Forest) has the highest accuracy score
* Tuned Random Forest Has The Best Accuracy Among All Algorithm
  + From all the evaluation matrix to predict hotel cancellation classification case, we see that Tuned Random Forest has the best accuracy when it comes to predicting hotel cancellation based on certain features (89.14 %)

### Implication on the business

* This model will allow hotel managers / revenue manager to take actions on bookings that's identified as "potentially going to be canceled", furthermore the development of these model should contribute to hotel revenue management.
* These prediction models enable hotel managers to mitigate revenue loss derived from booking cancellations and to mitigate the risks associated with overbooking (reallocation costs, cash or service compensations, and, particularly important today, social reputation costs). Booking cancellations models also allow hotel managers to implement less rigid cancellation policies, without increasing uncertainty. This has the potential to translate into more sales, since less rigid cancellation policies generate more bookings

**THANK YOU**