HOLIDAY PACKAGE PURCHASE PREDICTION

Sharath Pai

Date: 18th November 2023

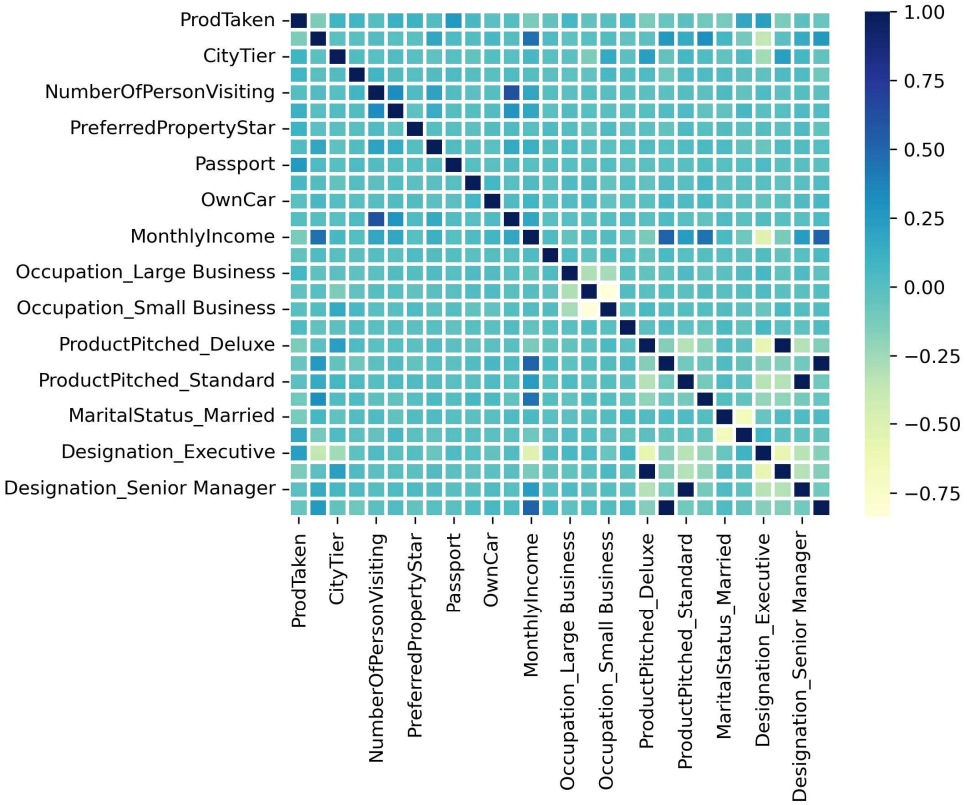
1. Problem Statement

Trips&Travels.com, a tours and travels company wants to improve its holiday package sales by using Exploratory Data Analytics and different Supervised Machine Learning algorithms for identifying the potential customers for a particular holiday package and target them by personalized marketing campaigns. However, the marketing cost was too high as customers were contacted randomly without prior knowledge regarding the available information about the package.

2. Customer Needs Assessment

The company has gathered information in the form of a dataset of which includes their occupation, designation, past purchases etc. Using Machine Learning, we can identify the customer behaviour and accordingly predict which customer is likely to purchase a holiday package. A product can be created out of this model for suggesting the appropriate package to customers by identifying the patterns out of the available dataset.

3. Target Specifications and Characterization



Based on the EDA, the company identifies the customers having designation as Executives as the most potent customers for their business model. Additionally the customers having a passport and are unmarried have a higher probability of purchasing the product.

4. Benchmarking

All major online travel agencies (OTAs) in India, utilize machine learning (ML) extensively to enhance their services and optimize their operations. Here is the case study of four of the major travel agencies using Machine Learning in their business models.

MakeMyTrip:

Makemytrip is among the largest travel MNCs in India. The agency uses ML to personalize travel recommendations, optimize pricing, and enhance customer service. For example, it analyzes customer pereferences to provide tailored suggestions for flights, hotels, holiday packages, and local experiences. The company also uses ML for varying prices based on factors such as demand, supply, pricing of rival markets, and customer behaviour. Additionally, Makemytrip utilizes AI powered virtual assistants and chatbots powered by ML to provide customer support all day, any day.

Kesari:

Kesari is a leading corporate travel management company in India. The company uses ML to optimize travel expenses, manage travel policies, and enhance traveler safety. For instance, Kesari uses ML to analyze travel patterns and identify opportunities to reduce travel expenses. The company also uses ML to enforce travel policies and ensure compliance with company regulations. Additionally, Kesari utilizes ML to track traveler locations and provide real-time alerts in case of emergencies.

Yatra:

Yatra is a popular online travel agency in India. The company uses ML to personalize travel recommendations, improve search optimization, and provide fraud detection. For example, Yatra uses ML to identify customer behaviour to provide tailored suggestions for flights, hotels, holiday packages, and local experiences. The company also utilizes ML algorithms to optimize search results and provide relevant and personalized search results. Additionally, Yatra employs ML models to identify fraudulent transactions and protect against financial losses.

5. Applicable Patents

Several technology patents can be utilized to enhance the business model of a holiday package dataset. These patents encompass various aspects of travel planning, customer personalization, and optimization strategies. Let's explore some relevant examples:.

i. Personalized Travel Recommendations:

Patent US8918416B2: "System and method to providepersonalized travel recommendations based on user preferences and behavior".

This patent describes a system that analyzes user preferences, travel history, and demographic data to generate personalized travel recommendations. It considers factors such as interests, budget, travel style, and past booking patterns to suggest tailored itineraries, accommodation options, and activities.

ii. Fraud Detection and Prevention:

Patent US9673222B2: "System and method for detecting and preventing travel fraud".

This patent describes a system that identifies fraudulent transactions in the travel industry using machine learning techniques. It analyzes payment patterns, user behavior, and device information to flag suspicious activities and prevent financial losses.

iii. Search Optimization and Relevance:

Patent US10291999B2: "System and method for optimizing search results in the travel industry".

This patent outlines a system that improves the relevance and personalization of search results for travel-related queries. It utilizes ML models to analyze user intent, past search behavior, and contextual information for providing tailored search results that align with individual preferences.

iv. Customer Segmentation and Targeted Marketing:

Patent US9483820B2: "System and method for segmenting customers and generating targeted marketing campaigns in the travel industry".

This patent describes a system that classifies customers into distinct segments based on their demographics, travel preferences, and past booking patterns. It uses this segmentation data to develop personalized marketing campaigns that target specific customer groups with relevant offers and promotions.

v. Risk Assessment and Credit Scoring for Travel Financing:

Patent US9837786B2: "System and method for assessing creditworthiness of customers in the travel industry".

6. Applicable Regulations

1) Cosumer Rights: Consumers have the right to accurate information about holiday packages, including cancellation policies, refund procedures, and dispute resolution mechanisms.
2) Data Privacy and Consent: Businesses must collect, store, and use customer data in compliance with data privacy regulations, such as GDPR and CCPA. Transparency in data handling practices and secure data storage are crucial.
3) Travel Regulations: Businesses must ensure that their holiday packages comply with visa requirements for the destinations offered. Providing clear information on visa procedures and assisting customers in obtaining necessary visas is essential.
4) Tax Regulations: Businesses may be required to collect and remit tourism taxes or other levies imposed by local or national governments to support tourism infrastructure and development.
5) Environmental Regulations: Businesses must adopt environmentally sustainable practices to minimize their carbon footprint and protect the environment in the destinations they operate.

7. Applicable Constraints

Space

Requires larger computational space depending on further data to be obtained for being able to use a number of machine learning algorithms and to be able to sustain the size of software packages required to create the product. Also requires the physical structures such as servers and bandwidth to be able to handle the traffic.

Budget

Requires substantial investments in data acquisition, infrastructure resources, hiring software and machine learning engineers, purchasing various softwares, marketing and advertising and some unforeseen expenses. These costs will need to be carefully managed to ensure the financial viability of the project.

Expertise

Requires business knowledge to be able to provide insights on the model and even suggest some additional factors which could impact the business model. Also the EDA performed must be performed and analysed appropriately to be able to interpret potential customer targets.

8. Business Model

The product presents a treasure trove of business opportunities in the travel and tourism industry. This rich data source, brimming with historical travel patterns, customer preferences, and market trends, holds immense potential for revolutionizing the way travel companies operate, enhance customer experiences, and drive revenue growth.

By leveraging the power of this dataset, travel companies can implement personalized

travel recommendations that cater to individual preferences and past travel behaviors, leading to increased customer satisfaction and higher conversion rates. Additionally, dynamic pricing optimization strategies can be developed, enabling companies to adjust prices based on recent trends, competitor pricing, and market dynamics, maximizing revenue while maintaining customer satisfaction.

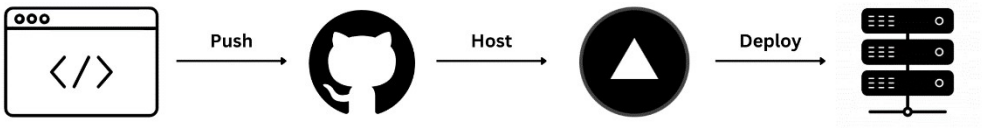
The dataset also facilitates targeted marketing campaigns and advertising placements, ensuring that promotions reach the right audience and deliver a higher return on investment. Furthermore, risk assessment and fraud detection models can be developed, minimizing financial losses and protecting revenue streams.

This product is not only a valuable tool for enhancing customer experiences and optimizing operations but also a powerful source of market insights. Travel companies can gain a deeper understanding of travel trends, customer preferences, and market dynamics, enabling them to make informed strategic decisions, develop innovative products, and tailor marketing initiatives effectively.

9. Concept Generation

The process involves creation of machine learning models by using various supervised learning algorithms. We train these algorithms on the training data and evaluate its performance on the test data. This helps us in differentiating between various models and conclude how our model performs on previously unseen data. Also, tuning the hyperparameters will ensure that we make use of appropriate parameters and will help us optimising the model. We'll analyze the performance of each model based on statistical measures such as Accuracy and ROC Area Under Curve score. We'll use the pickle library to save the best model and load the model into our website.

10. Concept Development



The product can be developed using Flask since it has an edge in terms of API creation. Flask is a lightweight Python backend framework used for building web applications. The model which was stored using the Pickle library can be used in Flask and it allows Pickle to serialize and deserialize objects and can be handy for loading user data, storing model parameters and caching data to improve user performance.

After the entire product is developed, it can be pushed into GitHub. We'll use Vercel which is a popular cloud platform that serves as a serverless deployment option for web

applications.
The GitHub repositories corresponding to the project can be triggered into Vercel and the Vercel cloud platform will deploy the website into its servers. The advantage of using GitHub here is that it integrates with Vercel and whenever we make some changes into the code, after pushing the code into the repository, the Vercel will automatically detect the changes and will deploy the website along with the changes.

11. Final Product Prototype

Frontend:

Personalized Recommendations: The frontend can display personalized holiday package recommendations based on user preferences, past travel patterns, and demographic factors. This can be achieved by using JavaScript frameworks like React or Angular to create dynamic and interactive user interfaces.

Visualizations and Data Representation: The frontend can effectively present insights and data derived from dataset. This can involve using charting frameworks like Chart.js for creating interactive visualization patterns that help users understand travel trends and patterns.

User Interaction and Feedback: The frontend facilitates user interaction with the product, allowing them to search for specific packages, filter results, and provide feedback on recommendations. This user interaction data can be collected and fed back into the backend for further analysis and improvement.

Backend:

Machine Learning Model Training: The backend is responsible for training and maintaining machine learning models that analyze the dataset and generate personalized recommendations. This involves selecting appropriate algorithms, training models on historical data, and evaluating model performance.

API Development and Integration: The backend develops APIs that expose the product's functionalities to the frontend. These APIs provide access to personalized recommendations, travel insights, and other data-driven features.

Data Security and Storage: The backend implements robust data security measures to protect sensitive user information and travel data. This involves encryption techniques, access control mechanisms, and secure data storage practices.

12. Product Details

Customers can utilize the product in various ways to enhance their travel planning experience. They can browse through a vast selection of holiday packages tailored to their preferences, budget, and destination interests. The product's search function enables them to filter packages based on specific criteria, making it easier to find the ideal getaway. Personalized recommendations provided by the product's machine learning algorithms help

customers discover new and exciting holiday packages that match their unique travel styles. These recommendations are based on past travel history, preferences, and demographic factors, ensuring a curated selection of options that cater to individual tastes. Detailed package information is readily available, providing customers with comprehensive descriptions, itineraries, pricing, inclusions, and exclusions. This kind of information enables them to make prehand decisions about their travel plans and choose the package that best suits their needs and budget.

The product allows for seamless comparison of different holiday packages, enabling customers to evaluate options side-by-side. They can compare factors such as price, duration, activities, and customer reviews, ensuring they make the most suitable choice for their travel requirements. Convenient booking directly through the product streamlines the process of securing holiday packages. The secure booking process ensures a smooth and hassle-free experience, allowing customers to finalize their travel arrangements with ease.

The following table shows the budget along with the team required for this model

Item	*Description*	*Estimated Cost (INR)*
Machine Learning	Hiring data scientists or machine learning engineers	Upto ₹1,500,000
Infrastructure Costs	Servers, storage, network bandwidth	Upto ₹1,000,000
Software Development	Developing the website and integrating machine learning models	Upto ₹2,000,000
Marketing and Promotion	Advertising, social media campaigns, and public relations	Upto ₹1,500,000
Contingency	Unforeseen expenses	Upto ₹1,000,000
Total		₹7,000,000

13. Code Implementation

The code was implemented in Jupyter Notebook and the tech stacks used were

NumPy: Numeric representation of data and performing mathematical operations

Pandas: Creation of data frames, data preprocessing

Scikit-Learn: Providing libraries used in machine learning algorithms, model evalutation, training data and tuning of hyperparameters.

Seaborn: Visualising the trends and relations between different variables

Matplotlib: Visualising and saving plots

Let's view our dataset

```
In [2]: df = pd.read_csv('Travel.csv', header=0)
```

```
In [3]: df.head()
```

Out[3]:

	CustomerID	ProdTaken	Age	TypeofContact	CityTier	DurationOfPitch	Occupation	Gender	NumberOfPersonVisiting	NumberOfFollowups	ProductPitched
0	200000	1	41.0	Self Enquiry	3	6.0	Salaried	Female	3	3.0	Deluxe
1	200001	0	49.0	Company Invited	1	14.0	Salaried	Male	3	4.0	Deluxe
2	200002	1	37.0	Self Enquiry	1	8.0	Free Lancer	Male	3	4.0	Basic
3	200003	0	33.0	Company Invited	1	9.0	Salaried	Female	2	3.0	Basic
4	200004	0	NaN	Self Enquiry	1	8.0	Small Business	Male	2	3.0	Basic

```
In [4]: df.shape
```

```
Out[4]: (4888, 20)
```

First, the data was explored and preprocessed to identify outliers, missing values and creating dummy variables for categorical data. After preprocessing, we proceed further towards model training.

We use different machine learning algorithms to assess the accuracy and the ROC AUC score of each algorithm.

Entire dataset

Logistic Regression on entire dataset

In [53]:
```
from sklearn.linear_model import LogisticRegression
X = df.drop("ProdTaken", axis=1)
Y = df['ProdTaken']

lm = LogisticRegression()
lm.fit(X,Y)
print(lm.intercept_, lm.coef_)
```

```
[-0.04303749] [[-2.86694082e-02  4.06180076e-01  2.29826840e-02 -8.45979181e-02
   3.77885410e-01  2.29460730e-01 -7.98007215e-02  6.86525750e-01
   5.89570966e-02 -1.81451861e-02 -8.06647196e-03 -1.46784714e-04
  -1.47107977e-01  9.23347486e-02 -9.59034339e-02 -4.40451437e-02
   3.97274457e-02 -3.50989907e-01  2.36693500e-02  6.51379531e-02
  -2.24637666e-02 -2.95357654e-01  4.12221334e-01  2.41601048e-01
  -3.50989907e-01  2.36693500e-02]]
```

```
E:\Anaconda\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [55]:
```
# Performance metrics
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score

# Predicting on test set
print(confusion_matrix(Y, lm.predict(X)))
print('Accuracy: ', accuracy_score(Y, lm.predict(X)))
print('ROC AUC Score: ', roc_auc_score(Y, lm.predict(X)))
```

```
[[3855  113]
 [ 679  241]]
Accuracy:  0.8379705400981997
ROC AUC Score:  0.6167393495792426
```

We get the overall accuracy on the entire dataset as 83.79%. We'll try to improve the accuracy and performance of the entire model by using test-train split on the individual models.

Test-Train Split

Test-Train Split

In [56]:
```
X = df.drop("ProdTaken", axis=1)
Y = df['ProdTaken']
```

In [57]:
```
from sklearn.model_selection import train_test_split
```

In [58]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

In [59]:
```
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(3910, 26) (978, 26) (3910,) (978,)
```

1) Logistic Regression

Logistic Regression

In [57]:
```
from sklearn.linear_model import LogisticRegression
logistic = LogisticRegression()
logistic.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, logistic.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, logistic.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, logistic.predict(X_test)))
```

```
[[789   9]
 [158  22]]
Accuracy:  0.8292433537832311
ROC AUC Score:  0.5554720133667502
```

2) Linear Discriminant Analysis

Linear Discriminant Analysis

In [58]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, lda.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, lda.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, lda.predict(X_test)))
```

```
[[778  20]
 [130  50]]
Accuracy:  0.8466257668711656
ROC AUC Score:  0.6263575605680868
```

Importing the GridSearchCV library for tuning the hyperparameters and standardising the variables

In [59]:
```
# Grid Search CV for tuning hyperparameters
from sklearn.model_selection import GridSearchCV
```

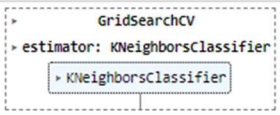
In [60]:
```
# Standardizing the variables
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_s= scaler.transform(X_train)
scaler = preprocessing.StandardScaler().fit(X_test)
X_test_s= scaler.transform(X_test)
```

3) K-Nearest Neighbors

K-Nearest Neighbors

In [64]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
params = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]}
tuning = GridSearchCV(knn, params)
tuning.fit(X_train_s, Y_train)
```

Out[64]:



In [65]:
```
tuning.best_params_
```

Out[65]:
```
{'n_neighbors': 1}
```

In [66]:
```
optimised_KNN = tuning.best_estimator_
```

In [67]:
```
# Predicting on test set
print(confusion_matrix(Y_test, optimised_KNN.predict(X_test_s)))
print('Accuracy: ', accuracy_score(Y_test, optimised_KNN.predict(X_test_s)))
print('ROC AUC Score: ', roc_auc_score(Y_test, optimised_KNN.predict(X_test_s)))
```

```
[[769  29]
 [ 28 152]]
Accuracy:  0.941717791411043
ROC AUC Score:  0.9040517961570593
```

4) Decision Tree

Decision Tree

In [ ]:
```
from sklearn import tree
class_tree = tree.DecisionTreeClassifier(max_depth=3)
params = {'min_samples_split': [1, 2, 3, 4, 5, 6, 7]}
tuning = GridSearchCV(class_tree, params)
tuning.fit(X_train, Y_train)
```

In [69]:
```
tuning.best_params_
```

Out[69]:
```
{'min_samples_split': 2}
```

In [70]:
```
optimised_tree = tuning.best_estimator_
```

In [71]:
```
# Predicting on test set
print(confusion_matrix(Y_test, optimised_tree.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, optimised_tree.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, optimised_tree.predict(X_test)))
```

```
[[787  11]
 [147  33]]
Accuracy:  0.8384458077709611
ROC AUC Score:  0.5847744360902256
```

5) Bagging

Bagging

In [70]:
```
from sklearn.ensemble import BaggingClassifier
bagging = BaggingClassifier(estimator=class_tree, n_estimators=1000, bootstrap=True, n_jobs=-1, random_state=0)
bagging.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, bagging.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, bagging.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, bagging.predict(X_test)))
```

```
[[784  14]
 [144  36]]
Accuracy:  0.8384458077709611
ROC AUC Score:  0.5912280701754385
```

6) Random Forest

Random Forest

In [71]:
```
from sklearn.ensemble import RandomForestClassifier
randomForest = RandomForestClassifier(n_estimators=1000, n_jobs=-1 ,random_state=0)
randomForest.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, randomForest.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, randomForest.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, randomForest.predict(X_test)))
```

```
[[791   7]
 [ 67 113]]
Accuracy:  0.9243353783231084
ROC AUC Score:  0.8095029239766082
```

7) Gradient Boost

Gradient Boost

In [72]:
```
from sklearn.ensemble import GradientBoostingClassifier
gbm = GradientBoostingClassifier(learning_rate=0.02, n_estimators=1000, max_depth=1)
gbm.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, gbm.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, gbm.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, gbm.predict(X_test)))
```

```
[[789   9]
 [148  32]]
Accuracy:  0.8394683026584867
ROC AUC Score:  0.583249791144528
```

8) Adaptive Boost

Adaptive Boost

In [73]:
```
from sklearn.ensemble import AdaBoostClassifier
adaboost = AdaBoostClassifier(learning_rate=0.05, n_estimators=5000)
adaboost.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, adaboost.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, adaboost.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, adaboost.predict(X_test)))
```

```
[[773  25]
 [128  52]]
Accuracy:  0.843558282208589
ROC AUC Score:  0.6287802840434419
```

9) XG Boost

XG Boost

In [74]:
```
import xgboost as xgb
xgboost = xgb.XGBClassifier(max_depth=5, n_estimators=10000, learning_rate=0.3, n_jobs=-1)
xgboost.fit(X_train, Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, xgboost.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, xgboost.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, xgboost.predict(X_test)))
```

```
[[788  10]
 [ 47 133]]
Accuracy:  0.941717791411043
ROC AUC Score:  0.8631787802840435
```

10) Stacking Classifier

Stacking Classifier

In [71]:
```
from sklearn.ensemble import StackingClassifier
estimators = [
    ('randomForest', RandomForestClassifier(n_estimators=1000, n_jobs=-1)),
    ('xgboost', xgb.XGBClassifier(max_depth=5, n_estimators=10000, learning_rate=0.3, n_jobs=-1))
]

stacking = StackingClassifier(estimators=estimators)
stacking.fit(X_train,Y_train)

# Predicting on test set
print(confusion_matrix(Y_test, stacking.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, stacking.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, stacking.predict(X_test)))
```

```
[[785  13]
 [ 33 147]]
Accuracy:  0.9529652351738241
ROC AUC Score:  0.9001879699248121
```

11) Radial Kernel

Radial Kernel

In [72]:
```
from sklearn import svm
svc = svm.SVC(kernel='rbf')
params = {'C':(0.01,0.05, 0.1, 0.5, 1, 5, 10, 50),
          'gamma':(0.001, 0.01, 0.1, 0.5, 1)}

tuning = GridSearchCV(svc, params, n_jobs=-1, cv=3, verbose=1, scoring='accuracy')
tuning.fit(X_train, Y_train)
```

```
Fitting 3 folds for each of 40 candidates, totalling 120 fits
```

Out[72]:
```
GridSearchCV(cv=3, estimator=SVC(), n_jobs=-1,
             param_grid={'C': (0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50),
                         'gamma': (0.001, 0.01, 0.1, 0.5, 1)},
             scoring='accuracy', verbose=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [73]:
```
tuning.best_params_
```

Out[73]:
```
{'C': 5, 'gamma': 0.1}
```

In [74]:
```
optimised_svc = tuning.best_estimator_
```

In [75]:
```
# Predicting on test set
print(confusion_matrix(Y_test, optimised_svc.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, optimised_svc.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, optimised_svc.predict(X_test)))
```

```
[[798   0]
 [ 89  91]]
Accuracy:  0.908997955010225
ROC AUC Score:  0.7527777777777778
```

12) Polynomial Kernel

Polynomial Kernel

In [76]:
```
svcP = svm.SVC(kernel='poly')
params = {'degree':(1, 2, 3, 4),
          'C':(0.001, 0.01, 0.1, 0.5, 1, 5, 10)}

tuning = GridSearchCV(svcP, params, n_jobs=-1, cv=3, verbose=1, scoring='accuracy')
tuning.fit(X_train, Y_train)
```

```
Fitting 3 folds for each of 28 candidates, totalling 84 fits
```

Out[76]:
```
GridSearchCV(cv=3, estimator=SVC(kernel='poly'), n_jobs=-1,
             param_grid={'C': (0.001, 0.01, 0.1, 0.5, 1, 5, 10),
                         'degree': (1, 2, 3, 4)},
             scoring='accuracy', verbose=1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [77]:
```
tuning.best_params_
```

Out[77]:
```
{'C': 0.001, 'degree': 1}
```

In [78]:
```
optimised_svcP = tuning.best_estimator_
```

In [79]:
```
# Predicting on test set
print(confusion_matrix(Y_test, optimised_svcP.predict(X_test)))
print('Accuracy: ', accuracy_score(Y_test, optimised_svcP.predict(X_test)))
print('ROC AUC Score: ', roc_auc_score(Y_test, optimised_svcP.predict(X_test)))
```

```
[[798   0]
 [180   0]]
Accuracy:  0.8159509202453987
ROC AUC Score:  0.5
```

13) Linear Kernel

Linear Kernel

```
In [80]: svcL = svm.SVC(kernel='poly')
         params = {'C':(0.001, 0.01, 0.1, 0.5, 1, 5, 10)}

         tuning = GridSearchCV(svcL, params, n_jobs=-1, cv=3, verbose=1, scoring='accuracy')
         tuning.fit(X_train, Y_train)

         Fitting 3 folds for each of 7 candidates, totalling 21 fits

Out[80]: GridSearchCV(cv=3, estimator=SVC(kernel='poly'), n_jobs=-1,
                      param_grid={'C': (0.001, 0.01, 0.1, 0.5, 1, 5, 10)},
                      scoring='accuracy', verbose=1)
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [81]: tuning.best_params_

Out[81]: {'C': 0.001}

In [82]: optimised_svcL= tuning.best_estimator_

In [83]: # Predicting on test set
         print(confusion_matrix(Y_test, optimised_svcL.predict(X_test)))
         print('Accuracy: ', accuracy_score(Y_test, optimised_svcL.predict(X_test)))
         print('ROC AUC Score: ', roc_auc_score(Y_test, optimised_svcL.predict(X_test)))

         [[798   0]
          [180   0]]
         Accuracy:  0.8159509202453987
         ROC AUC Score:  0.5
```

Here's a table providing the summary of the performance metrics of all the models

Model	*Accuracy (%)*	*ROC AUC Score (%)*	*Hyperparameters*
Entire dataset	83.79	61.67	
Logistic Regression	82.94	55.54	
LDA	84.66	62.63	
KNN	94.17	90.40	k=1
Decision Trees	83.84	58.47	min sample split=2, max depth=3
Bagging	83.84	59.12	no. of estimators=100
Random Forest	92.43	90.05	no. of estimators=100
Gradient Boost	83.94	58.32	learning rate=0.02, no. of estimators=1000, max_depth=1
Adaptive Boost	84.35	62.87	learning rate=0.05, no. of estimators=5000
XG Boost	94.17	86.31	learning rate=0.5, no. of estimators=10000, max_depth=5
Stacking Classifier	95.50	90.14	
Radial Kernel	90.89	75.27	cost=5, gamma=0.1
Polynomial Kernel	81.59	50	cost=0.001, degree=1
Linear Kernel	81.59	50	cost=0.001

Here, the Stacking Classifier provides the best accuracy on the test set as 90.89%. Stacking Classifier is the combination of two of our best ensembling models: i.e. XG Boost and Random Forest. We use the results obtained from the Stacking Classifier model into our product.

GitHub Link to the entire code: https://github.com/Sharath1036/holiday-package-purchase-prediction

14. Conclusion

The product provides an effective solution for personalized holiday package recommendations, enhancing customer satisfaction and increasing conversion rates. It utilizes machine learning models trained on historical data to accurately predict customer preferences and suggest tailored holiday packages that align with their interests and budget. By continuously analyzing user interactions and feedback, the website refines its recommendations and improves its overall effectiveness in helping customers discover and book their ideal holiday packages.

15. References

Dataset: https://www.kaggle.com/datasets/susant4learning/holiday-package-purchase-prediction

Case study of companies: https://analyticsindiamag.com/

Patents: https://patents.google.com

Government Laws and Regulations: https://www.indiacode.nic.in/