Machine Learning

Travel Insurance Data

Southern Illinois University Edwardsville

Sagar kalauni

Abstract

This study focuses on the significance of travel insurance in providing security for both domestic and international travelers. The aim is to determine customer interest in purchasing travel insurance based on various factors such as age, employment type, education, income, family size, chronic diseases, frequent flying, and past travel experiences. The dataset includes 1887 rows of data representing different travelers, with 9 columns capturing diverse information. The study employs nine classification algorithms and ensemble methods, including logistic regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbors (KNN), Decision Tree, Bagging, Boosting, Gradient Boosting, Random Forest, and Support Vector Machine. After thorough evaluation, Random Forest emerges as the top performer with an accuracy of 84.8% in predicting whether a customer would purchase travel insurance. The recommended model is deemed advantageous for insurance companies, providing insights to target specific individuals effectively and optimize profits.

ML Project STAT 562

Sagar Kalauni

2023-11-18

Table of Contents

Overview

This project involves the analysis of data that is related to a travel insurance package that a tour and travels company is offering to its customers which also includes COVID — 19 coverage as a new insurance package. The company wants to know which of its customers would be interested in buying the insurance package which was(i.e insurance package) initially offered to some of its customers in 2019.

The dataset consists of several predictor variables that are related to its customers and one target variable, TravelInsurance, which represents whether or not the customer will buy travel insurance package. Predictor variables includes: Age- Age of the customer, Employment Type - The sector in which customer is employed, GraduateOrNot - Whether the customer is college graduate or not, AnnualIncome - The yearly income of the customer, FamilyMembers - Number of members in customer's family, ChronicDiseases - Whether the customer suffers from any major disease, FrequentFlyer - Derived data based on customer's history of booking air tickets on at least 4 different instances in the last 2 years And EverTravelledAbroad - Has the customer ever travelled to a foreign country. The data we have for the analysis is of almost 2000 of its customer, extracted from the performance and sales figures of the insurance package during the year 2019.

This project was the part of my academic course STAT 562 (Machine learning and classification Methods). The primary language for the analysis will be R programming as per the course requirement and since R is purely for statistics and data analysis, with graphs are nicer and more customizable in R compared to python. For this analysis purpose we are assuming data were unbiased while collection but only needing some cleaning before analysis i.e we will not question the biasedness of the data.

Objective

The objective of this project is to predict whether or not the customer will buy travel insurance package, based on factors included in the dataset that pertain to customer Age, Employment Type,GraduateOrNot, AnnualIncome, FamilyMembers, ChronicDiseases, FrequentFlyer and EverTravelledAbroad.

By predicting whether or not the customer will buy a travel insurance package, management at the tour and travels company can optimize their or maybe focus on specific advertising campaigns. Clear explanation is not given in future use of this analysis but we can easily assume that this analysis will be helpful for the company's future growth and development.

Review of the data source

The data that was used for this assignment (TravelInsuranceData.csv and TravelInsuranceTest.csv) was provided by Professor Dr. Bidi Qiang for the project purpose. The dataset contains 10 columns and 1,887 rows. The dataset does not contain any null values in any columns. Unnecessary columns were dropped (Column 1 was just a indices column) so we remained with 9 variables in the dataset among which 8 are Predictor variable and 1 target variable.

It is always a good idea to spend a good amount of time knowing more about your data because study of metadata of the data makes the data analysis process smooth and fine. In our dataset we have GraduateOrNot, FrequentFlyer, EverTravelledAbroad, EmploymentType as categorical variables and remaining as Numerical variable. We can also perform necessary datatype transformation if needed.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical step in the data analysis process. It involves examining and visualizing the dataset to summarize its main characteristics, often with the help of statistical graphics and other data visualization methods. Some common steps and techniques involved in Exploratory Data Analysis are:

1) Summary Statistics
2) Data Visualization
3) Outlier Detection
4) Correlation Analysis etc and more.

General Observation from EDA:

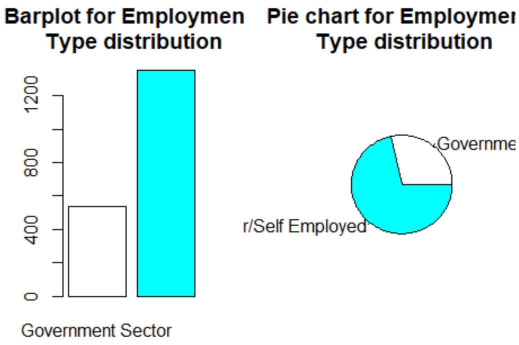
If you want the EDA insights right away, check out the 'Infer' section. Here, I'm letting the data speak for itself to reveal its story.

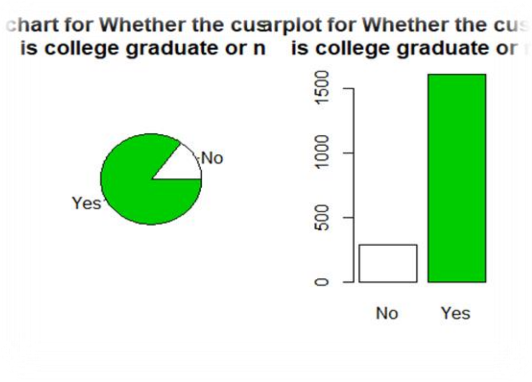
Looking at the summary statistics, Plots, and charts of the dataset we get the following general observation observation.

- Minimum and maximum age of the customer is 25 and 35 respectively and average age is around 30 (i.e 29.64)

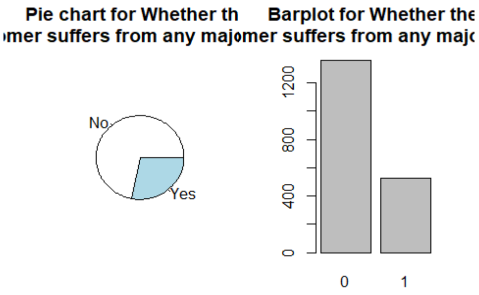
- Minimum and maximum annual income of the customer is 300,000 and 1,800,000 respectively and average salary is around 900,000

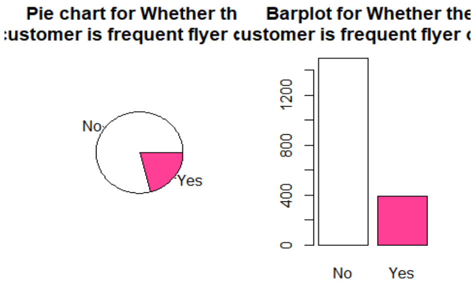
- Minimum and maximum number of family members in the customer's family is 2 and 9 respectively with average of around 5



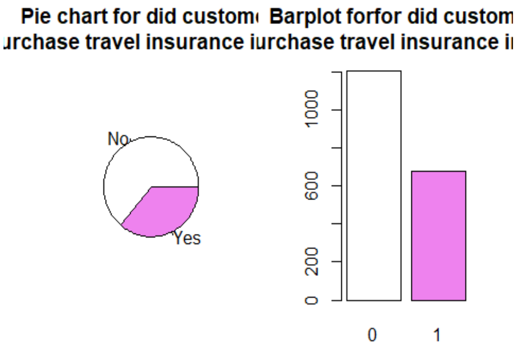


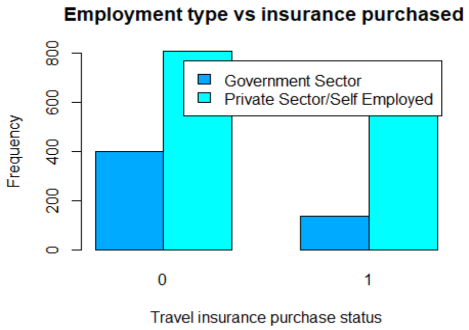
- ❖ The number of customers working in the Governmental sector are quite less as compared to customer working for the private sector or so-called self-employed. About only 535 customers work for government but 1352 for the privete sector among all the customers.
- ❖ Most of the customers were college graduates.
- ❖ Out of total 1887 customers, only 282 were customer without college degree



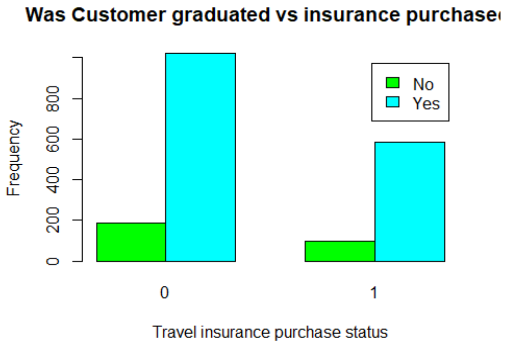


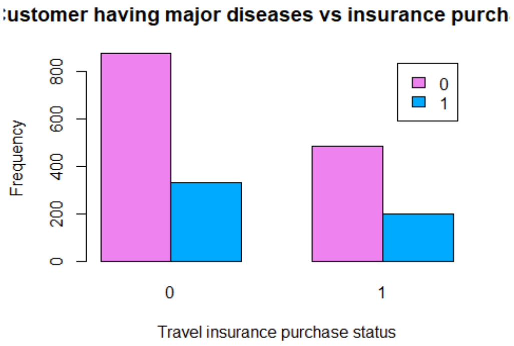
- ❖ Most of the customers do not suffer from any major disease.
- ❖ Out of total customers only 528 customers are suffering from major disease but 1359 of the customers were not suffering from any major diseases.
- ❖ Most of the customers are not frequent flyers, only 392 customers out of total customers are frequent flyers.





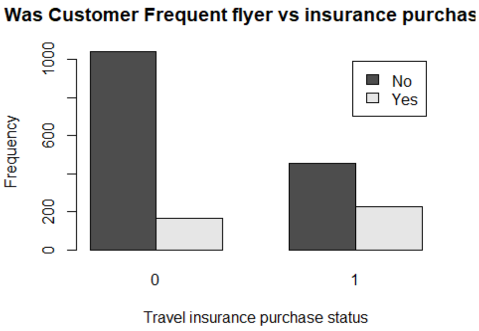
- ❖ Looking at the given data of 2019, most of the customers did not buy a travel insurance package. Only around 36% of the customer did purchase the travel insurance package back in 2019.

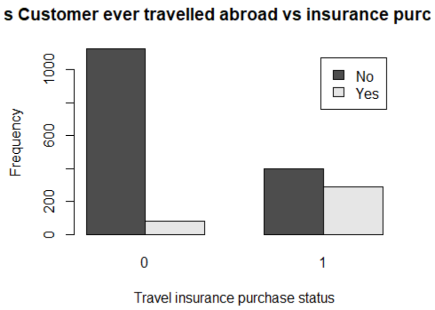




- ❖ Customers who are not graduated and not purchasing the travel insurance are more than customers who are graduated and purchasing insurance.

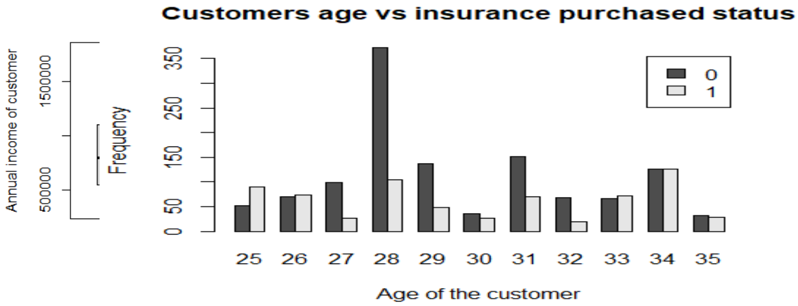
- ❖ Interestingly customers having some major disease and not purchasing the travel insurance (484) is more than customer having major disease and purchasing travel insurance (197).



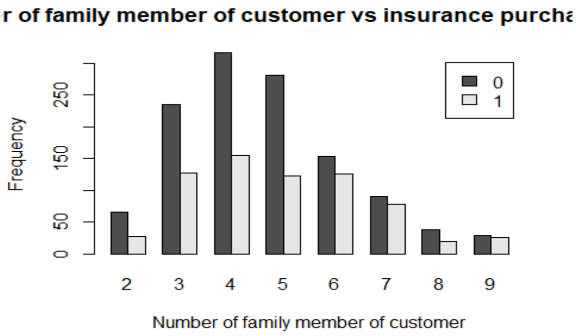


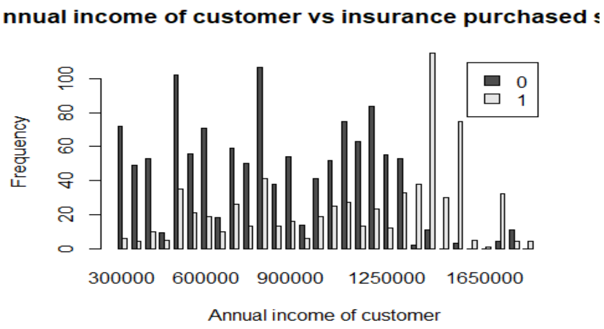
- ❖ Customers who purchase travel insurance and are frequent flyers (226) are more than customers who did not purchase travel insurance but are frequent flyers (166).

- Customers who have purchased travel insurance and have traveled abroad (286) are more than customers who did not purchase travel insurance but traveled abroad (81).
- The highest number of customers are of the the age 27.
- The distribution of Annual Income looks modified normal.
- Most of the customers have in general more than 4 family members in the family.



- The median income of the people who purchase travel insurance is more than the median income of those who did not purchase travel insurance in 2019.
- The median number of family members in the family for customers who purchase travel insurance is more than those who did not purchase the travel insurance in 2019.
- Customer of the age 34 has the highest number of travel insurance purchased.
- Customer at the age of 25 has more proportion of purchasing the travel insurance.
- In general, low age or high age customers are more likely to purchase travel insurance then middle aged one.





- Customers with number of family member 4 are the one that has the highest number of travel insurance purchased.

- ❖ Customers which lie in the high salary range are almost sure to purchase the travel insurance and customers having low salary are almost sure to of not purchasing the travel insurance. (here low means < 600000 and High means > 1250000)

Inference From EDA

- ✓ Looks like if the customer is working in the private sector, chances of taking travel insurance are little high compared to customer working for the government sector. The reason behind this may be any like government job has less pay, or government job has their own insurance and does not need travel insurance.
- ✓ The conversion rates for "Yes" and "No" responses on Chronic Diseases are quite similar, with a slightly higher rate for "Yes," suggesting that Travel Insurance offerings may be less relevant to individuals with chronic health conditions.
- ✓ People who travel a lot ("Frequent Flyers") are more likely to buy travel insurance. This could be because they might need it more due to their travel habits, and the insurance options offered, like coverage for multiple trips and flight-related issues, match their needs.
- ✓ Customers who have "Ever Travelled Abroad" are likely to buy travel insurance, possibly because they are more aware of the risks and challenges linked to international travel.
- ✓ Customers with Higher Annual Income are more likely to purchase the travel insurance, this could be possibly because the Insurance plan provided by the tour and travel company is really expensive for normal people to afford.

Machine Learning Models

Now that we understand our data well, our goal is to predict whether a customer will buy travel insurance based on their profile. To do this, we'll use different machine learning models for prediction. There are many models available, but we can't use them all due to time constraints. We'll select a few models, check their accuracy, and only predict our test set using the best-performing model among our choices.

In simple terms, machine learning is like teaching computers to learn from data without being explicitly programmed. This helps them get better at tasks over time as they see more data. In this paper we will use some ensemble methods and machine learning methods like: Logistic Model, LDA, QDA, KNN- classifier, Decision Tree, Random Forest, Support Vector machine (SVM), Bagging, Boosting, gradient Boosting.

For each model we will try to look at different- different parameters to compare the models: Most commonly we will focus on Accuracy of the model, Precision and Recall.

Where Accuracy means how accurate is your model in predicting the values. Accuracy is given by : $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ Where

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative.

Similarly Recall and Precision are given as $Recall = \frac{TP}{TP+FN}$ and $Precision = \frac{TP}{TP+FP}$.

Another important thing to mention is that we split our main bunch of data into two parts: 70% for training our model and 30% for testing it. We teach our model with the training set and check how well it is doing with the testing set. It's crucial to highlight that when I mention the "test set" in the model, I'm talking about the part we use to check how accurate the model is, not the one we'll predict in the end. For predictions, we'll only use a separate set that is given aside.

Feature Selection

Before diving into different machine learning models, it's smart to figure out which variables matter the most and which ones don't really help in predicting whether someone will get travel insurance. There are some cool methods in logistic regression, like forward selection, backward selection, and LASSO Regression (L1 Regularization), that can help us decide which features are essential. In this paper, I used backward selection, and the cool thing is that it gave us the same important features as other methods like gradient boosting or decision trees. The Important features are:

- ✓ Age
- ✓ AnnualIncome
- ✓ FamilyMembers
- ✓ FrequentFlyerYes
- ✓ EverTravelledAbroadYes

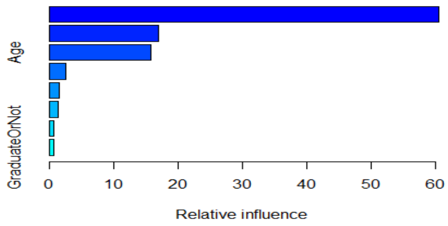


Fig: - Showing variable importance Gradient boosting

ChronicDiseases1, GraduateOrNotYes, Employment.TypePrivate Sector/Self Employed are not that too important for customer in purchasing the travel insurance.

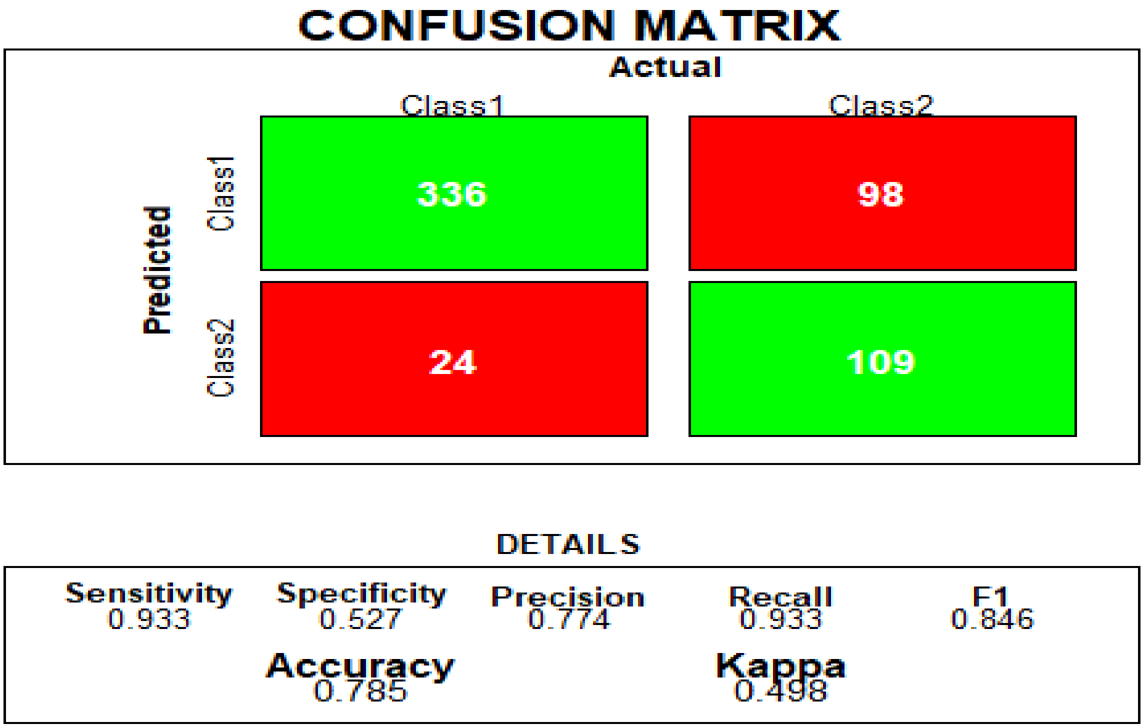
The above figure is with the lowest learning rate 0.001(shrinkage parameter) that also show same if learning rate is little increased that there 5 variable are most important as suggested by logistic model.

The best part? When we run the analysis with these selected features, the change in accuracy is only a little bit in most af methods (you can check the R code for details). In my analysis I have done all models with selected features and with all features too. But here I will only compare with all features models because adding all will make paper lengthy and confusing, but you are always welcome to check the code for how the model performs with selected feature only.

Logistic Model

In the world of predicting things, the logistic model is like the main character. It's super useful for dealing with situations where the answer can only be "Yes" or "No." (i.e binary) This model is really good at figuring out the chances of different outcomes. It helps us understand how likely certain events are to happen, making it great for decision-making.

Below is the results of predictions made by the logistic model in our test data set.

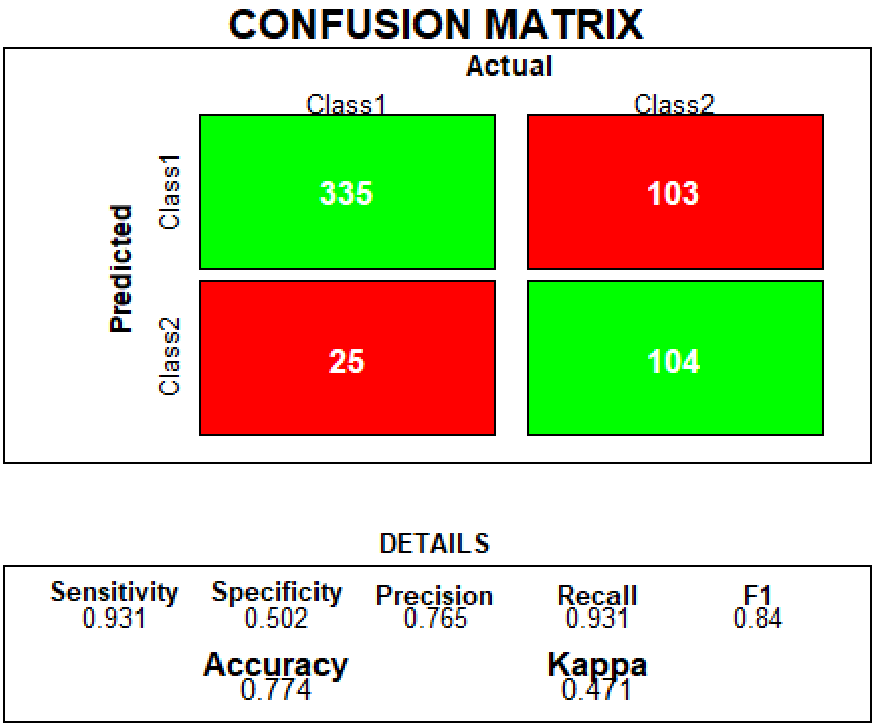


Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a method that helps us understand what features make different groups in our data unique, acting like a savvy detective in the world of predictions. Unlike logistic regression, which is excellent for two-category outcomes, LDA is particularly handy when we're dealing with more than two groups. It's like having a detective who can easily distinguish not only between apples and oranges but also bananas. LDA excels in scenarios where there are multiple categories, offering a unique advantage over logistic regression in providing insights into distinctions among various groups in our data.

```
lda.out=lda(TravelInsurance ~ Age + Employment.Type + GraduateOrNot + AnnualI
ncome + FamilyMembers +
              ChronicDiseases + FrequentFlyer + EverTravelledAbroad, data = t
rain_data)
```

We train our model using the training dataset and test our model using the test dataset, and the result of our model in our test data set is given below:



CONFUSION MATRIX

Predicted \ Actual	Class1	Class2
Class1	335	103
Class2	25	104

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.931	0.502	0.765	0.931	0.84

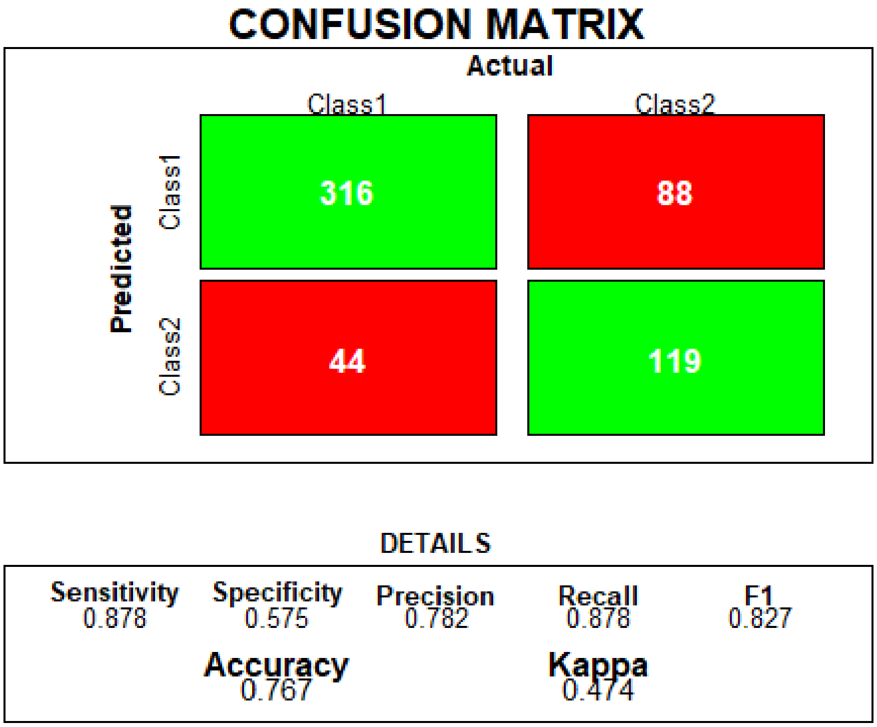
Accuracy	Kappa
0.774	0.471

Quadratic Discriminant Analysis

Quadratic Discriminant Analysis (QDA) is another detective in our prediction toolkit, working alongside Linear Discriminant Analysis (LDA). While LDA is like a detective focusing on finding features that make different groups stand out in a linear fashion, QDA takes it a step further. QDA considers not just linear relationships but also quadratic ones, allowing for more flexibility in capturing complex patterns in the data. This means it can handle situations where the distinctions between groups are not just straight lines but involve curves or bends. It's like having a detective that can navigate through more intricate clues. However, unlike logistic regression, which is great for binary outcomes, QDA is especially beneficial when we're dealing with multiple groups, providing a versatile approach to understanding and predicting complex relationships within our data.

It is slightly freer than LDA but still restricted than KNN-Classifier. We will talk about knn classifier next.

We train our model using the training dataset and test our model using the test dataset, and the result of our model in our test data set is given below:



CONFUSION MATRIX

Predicted \ Actual	Class1	Class2
Class1	316	88
Class2	44	119

DETAILS

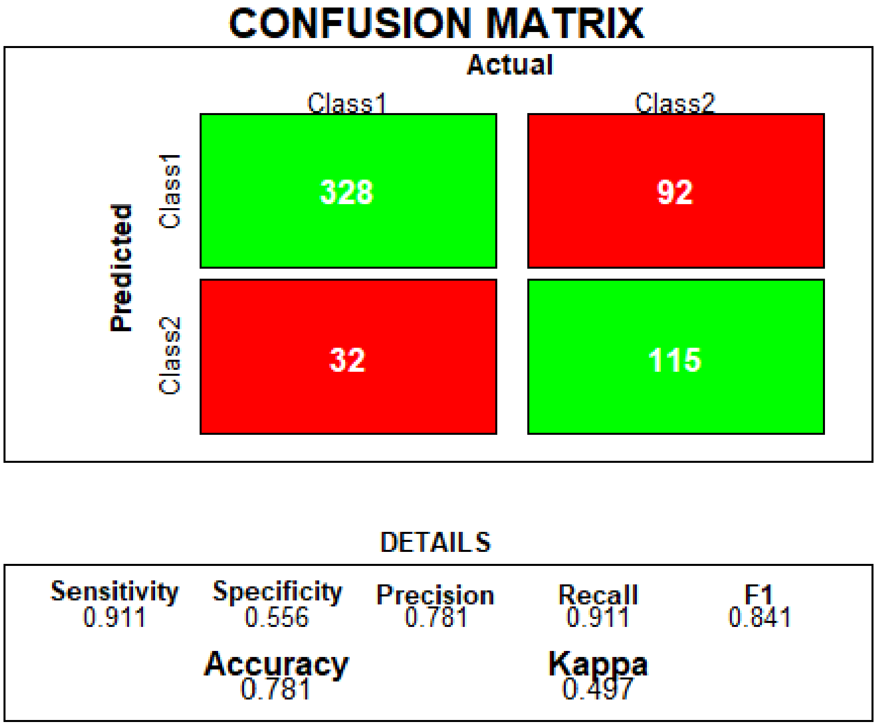
Sensitivity	Specificity	Precision	Recall	F1
0.878	0.575	0.782	0.878	0.827

Accuracy	Kappa
0.767	0.474

KNN- Classifier

K-Nearest Neighbors (KNN) is like a friendly neighbor in our predicted neighborhood. It's a classification algorithm that helps us predict outcomes based on the characteristics of nearby data points. Imagine each data point in our dataset as a house in a neighborhood. KNN looks at the features of a new data point and checks which existing data points are its closest neighbors. It then decides the category of the new point based on what most of its nearby neighbors belong to. Unlike linear or quadratic detectives, KNN doesn't assume specific patterns but instead relies on the collective wisdom of its nearest neighbors. It's a versatile approach that doesn't make many assumptions about the data structure. While it may not be as interpretable as logistic regression or LDA, KNN shines when dealing with complex patterns and is especially handy when we don't have a clear understanding of the underlying relationships in our data.

We did model tuning and cross-validation to find the best value of K that gives the maximum accuracy. We found out that for K=13, model has the maximum accuracy. We used 10 different K-values to check the model and found out K=13. With this we fit our model and predict our test data set and the results of the prediction are as below


CONFUSION MATRIX

Predicted	Actual: Class1	Actual: Class2
Class1	328	92
Class2	32	115

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.911	0.556	0.781	0.911	0.841

Accuracy	Kappa
0.781	0.497

```
set.seed(123)
knn_model= train(
  TravelInsurance ~.,
  data = train_data,
  method = "knn",
  tuneLength=10,
  trControl = trainControl(method = "cv", number = 10),
  preProcess = c("center", "scale")
)
```

Decision Tree

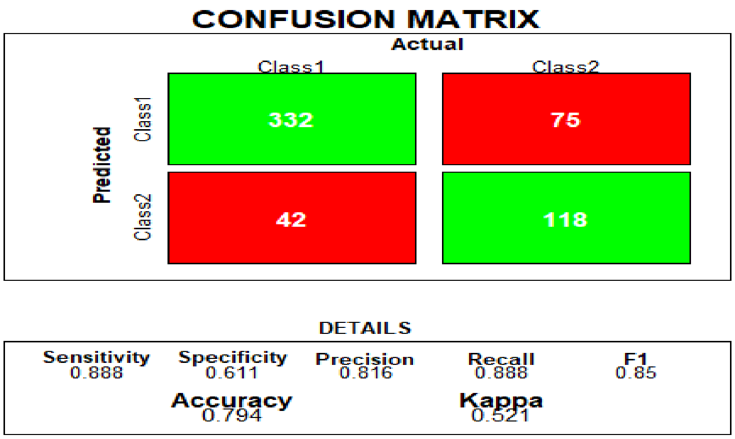
A Decision Tree is like a wise storyteller in the land of predictions, making decisions based on a sequence of questions. Picture it as a tree where each branch represents a question, and each leaf, an answer. It works by asking a series of questions about the features of our data, leading to different outcomes. Like a detective with a flowchart, a Decision Tree learns from the data to create this branching structure, making it a great visual aid for understanding decision-making. Unlike logistic regression or LDA, a Decision Tree doesn't assume linear or quadratic relationships. It adapts to the data's natural structure, making it valuable when dealing with complex scenarios. While it might not be as transparent as logistic regression, a Decision Tree's strength lies in its ability to handle intricate patterns and provide a clear, visual representation of decision paths in our data.

In our exploration of decision trees, we conducted experiments using both pruned and unpruned trees to predict our data. Surprisingly, we did observe a substantial difference in test accuracy between the two tree variants. These results suggest that, in the context of our dataset and problem, the performance gain achieved through pruning is high.

```
#Unpruned tree
tree.d=tree(TravelInsurance~.,Insurance_data,split="gini",subset=train) # exc
ept TravelInsurance all other variables in the data set are be considered as
predictors.
```

```
plot(tree.d)
```

We train our model using the training dataset and test our model using the test dataset, and the result of our model in our test data set is given below:



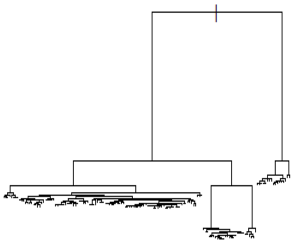


Fig: -Unpruned Tree
Unpruned Tree

Pruned Tree

```
# Pruned Tree

set.seed(12312)
cv.d=cv.tree(tree.d) # using deviance as a criteria for the cross-validation
plot(cv.d$size, cv.d$dev, type = "b") # Since we have used deviance as our cr
iteria for the cross-validation,
```

```
set.seed(12312)
prune.d=prune.tree(tree.d, best =6)


plot(prune.d)
text(prune.d)
```

Since we want a tree with minimum deviance and at the same time less complex tree, we found best=6 for pruning the tree. The result of pruned tree is shown in the figure below

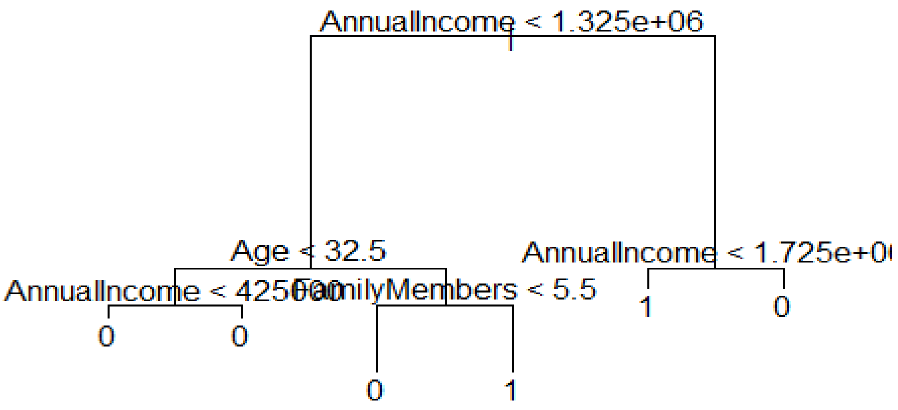
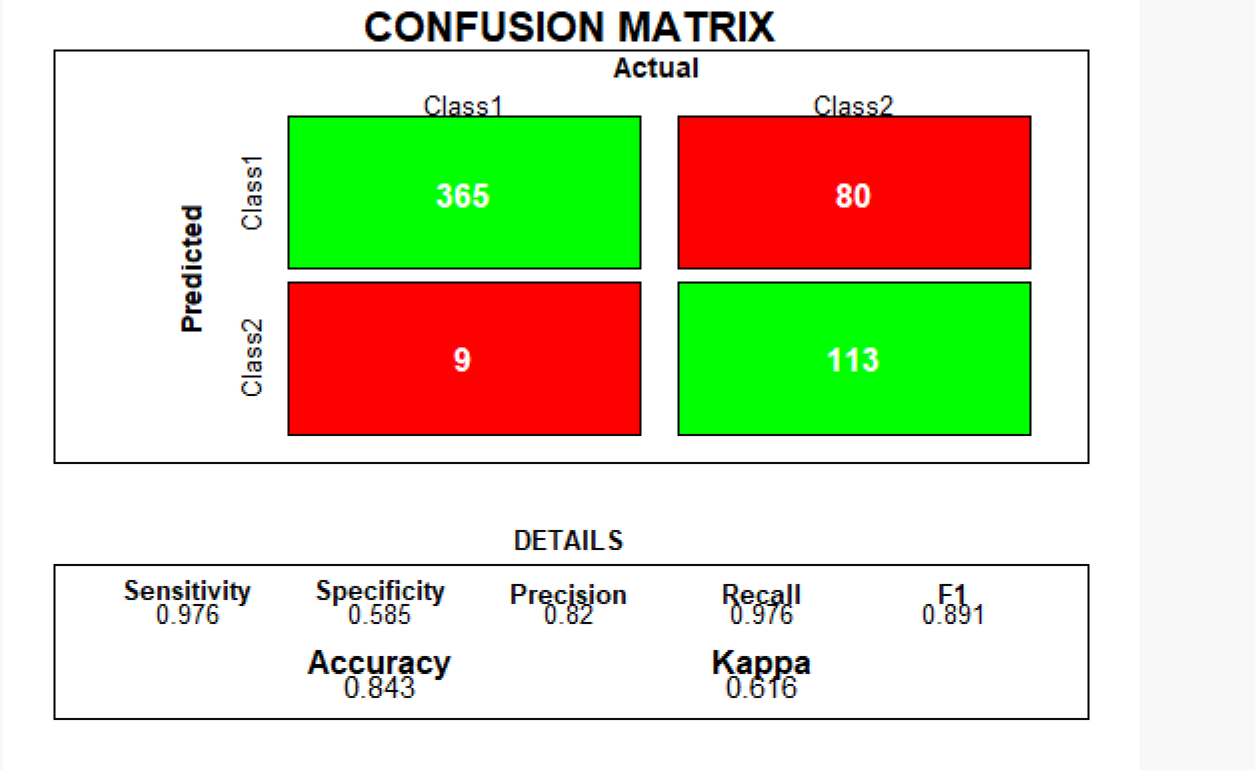


Fig: - Pruned Decision Tree

SO, from this pruned tree also, we can say that annual income is the key factor that will determine whether the customer will buy the travel insurance or not.



CONFUSION MATRIX

Actual

Predicted	Class1	Class2
Class1	365	80
Class2	9	113

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.976	0.585	0.82	0.976	0.891

Accuracy	Kappa
0.843	0.616

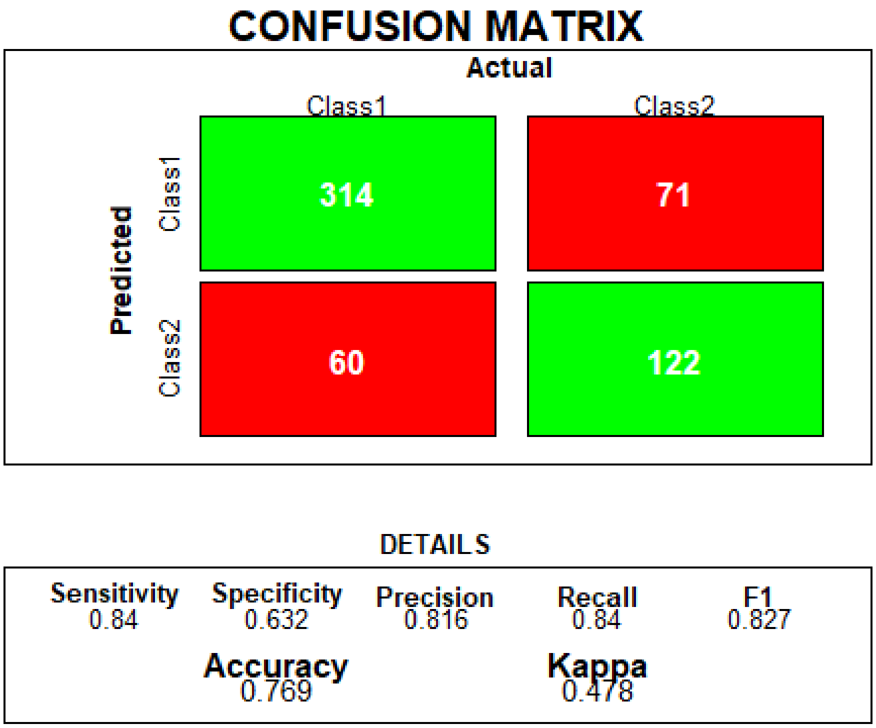
Bagging (RF)

Bagging, or Bootstrap Aggregating, is like a team-building strategy for predictive models. Instead of relying on a single model, Bagging creates multiple copies of the same model but trains each on a slightly different subset of the data. It's like having a bunch of detectives investigate different aspects of a case. Bagging helps reduce the impact of outliers or noisy data by considering a variety of perspectives. The beauty lies in combining the predictions of all these models, and it's particularly powerful when used with decision trees – this ensemble of trees is commonly known as a Random Forest. Unlike a solo detective (single decision tree), Bagging's team approach makes predictions more robust, accurate, and resistant to overfitting. It's a smart strategy to improve the overall performance of our predictive models, adding a touch of teamwork to the world of predictions.

When you're building a decision tree, you have a choice about how many features (or factors) to look at when deciding how to split the data. If you use all the features available, it's called bagging. But if you want to add a bit of randomness to make the tree more versatile, you can randomly pick a smaller number of features for each split. When you do this, your tree becomes part of something we call a Random Forest. We'll dive into Random Forest in more detail shortly.

```
# begging means talking all the feature variable for mtry
beg.Insurance_data=randomForest(TravelInsurance~., data=Insurance_data, subse
t= train, mtry=8, importance=TRUE)
beg.Insurance_data
```

When we train our model with the training data and use that model to predict our test data, we obtain the following results.



Random Forest

Random Forest is like a wise forest filled with decision-making trees, each contributing to a collective prediction. Think of it as having many detectives working together, each offering its unique insights. In this strategy, we create a bunch of decision trees using Bagging (Bootstrap Aggregating), where each tree gets trained on a different subset of the data. This diversification helps the forest handle various aspects of our predictive puzzle. When we need a prediction, each tree casts its vote, and the outcome with the most votes becomes the final prediction. It's like having a council of detectives sharing their opinions, resulting in a more accurate and reliable decision. Random Forest is fantastic for handling complex patterns, minimizing overfitting, and enhancing the overall predictive power of our models in a team-based fashion.

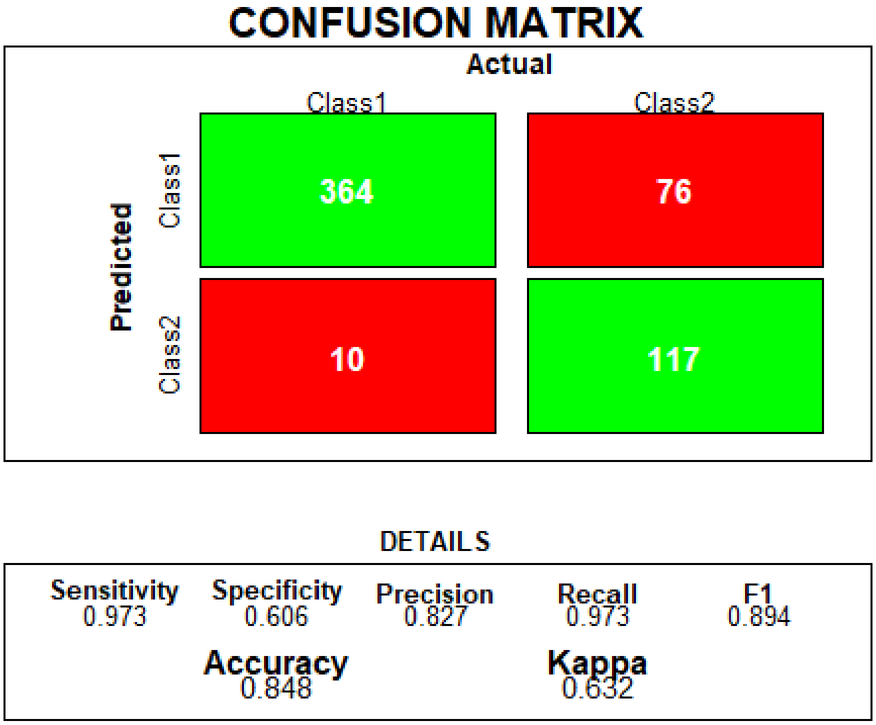
As we have discussed earlier also, here since we want more randomness in creating our decision tree, we did not use all the feature variables for splitting. As a general rule of thumb, we use mtry= square root of number of feature variables. We will tune this hyperparameter and choose that model for which Out-of-Bag (OBB) estimate of error rate is minimum.

Here since my number of feature variables are less, I tried almost all possible mtry values.

It seems that the Random Forest with mtry=2 gives us the lowest OOB error(17.88%) from your ouput.

```r
set.seed(12312)
# trying m=2
rf.Insurance_data_2=randomForest(TravelInsurance~., data=Insurance_data, subs
et=train, mtry=2, importance=TRUE)
rf.Insurance_data_2 # lets take a look at the output
```

Now when we predict our test dataset with this model we obtain the following results



CONFUSION MATRIX

Predicted \ Actual	Class1	Class2
Class1	364	76
Class2	10	117

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.973	0.606	0.827	0.973	0.894

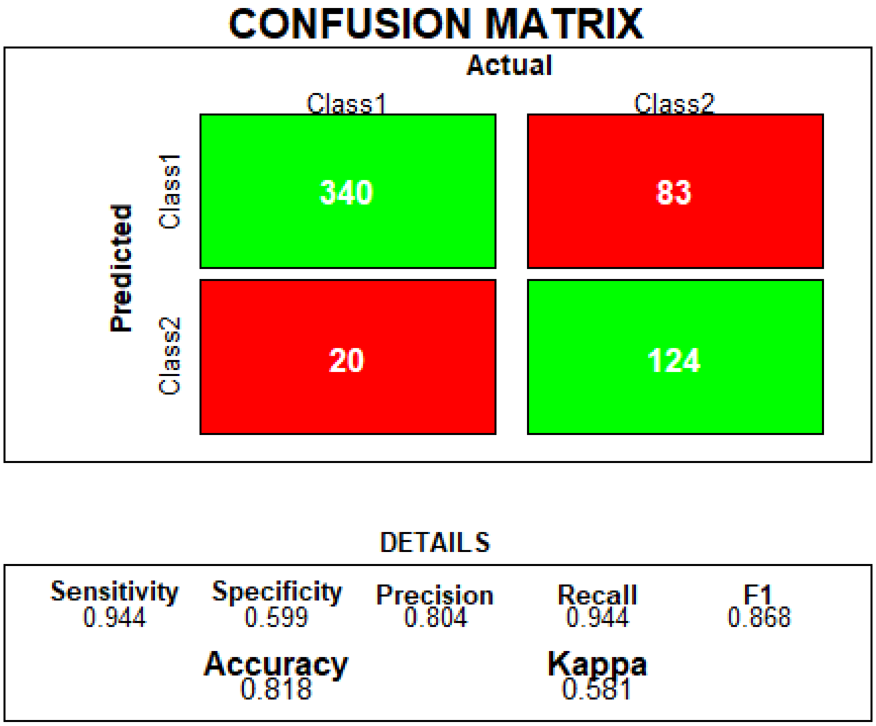
Accuracy	Kappa
0.848	0.632

Boosting (with Shallow decision Tree)

Boosting is like a coach turning a group of novice detectives into an elite team, refining their skills over time. Instead of creating many independent models, boosting builds a series of shallow decision trees, each learning from the mistakes of the previous one. It's like detectives learning from their past investigations and improving with each case. Boosting emphasizes the importance of the data points that were tricky for the previous trees, making the team collectively stronger. This approach is particularly effective when using shallow decision trees, where each tree is like a quick thinker focusing on specific details. The boosted team becomes more adaptable and better at handling complex patterns in the data, making it a potent strategy for enhancing predictive accuracy. I made 50 different shallow trees to make a decision.

```r
# Trying the boosting model
boosting.model= boosting(TravelInsurance~., data = train_data, mfinal = 50)
```

When we predict our test data set with this model we obtain the following results



CONFUSION MATRIX

Predicted	Actual Class1	Actual Class2
Class1	340	83
Class2	20	124

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.944	0.599	0.804	0.944	0.868

Accuracy	Kappa
0.818	0.581

Support Vector Machine (SVM)

A Support Vector Machine (SVM) is one of the best machine learning methods in the world of predictions, drawing clear lines between different categories. It's excellent for scenarios where we want to separate things into distinct groups. Picture it as finding the best fence to separate apples from oranges. The SVM looks for the optimal line (or boundary) that maximizes the gap between different groups, making it a master of precision. Unlike decision trees, SVM doesn't rely on the entire dataset but focuses on crucial data points, called support vectors. It's like identifying key witnesses in a case. SVM is especially useful when the data is a bit messy and doesn't follow a clear pattern. Its ability to create well-defined boundaries makes it one of the best in classifying and predicting outcomes accurately.

Support Vector Machines (SVMs) come with various hyperparameters that allow fine-tuning and optimization of the model. Here's a brief overview:

1. Kernel (Kernel Function):

The choice of kernel determines the type of decision boundary the SVM creates. Common kernels include linear, polynomial, and radial basis function (RBF). Selecting the appropriate kernel depends on the complexity of the data. Here in this paper, we will try to make our model for all these different types of kernels and check their accuracy

2. C (Regularization Parameter):

The regularization parameter, denoted as C, controls the trade-off between achieving a smooth decision boundary and classifying training points correctly. A smaller C encourages a broader margin but may misclassify points, while a larger C prioritizes correct classifications but might result in a narrower margin. We will also tune this parameter in our models and try to find the value of C, for which our model has the highest accuracy.

3. Gamma (Kernel Coefficient):

Applicable in non-linear kernels, gamma defines how far-reaching the influence of a single training point is. A smaller gamma implies a more widespread influence, whereas a larger gamma confines influence to nearby points, resulting in a more intricate decision boundary. . We will also tune this parameter in our models and try to find the value of gamma, for which our model has the highest accuracy.

Also when we are developing SVM model we have standardized our data set

A) Linear kernel

We are using cross-validation to find the best value of C from the range: cost=seq(0.01, 10, length.out = 20).

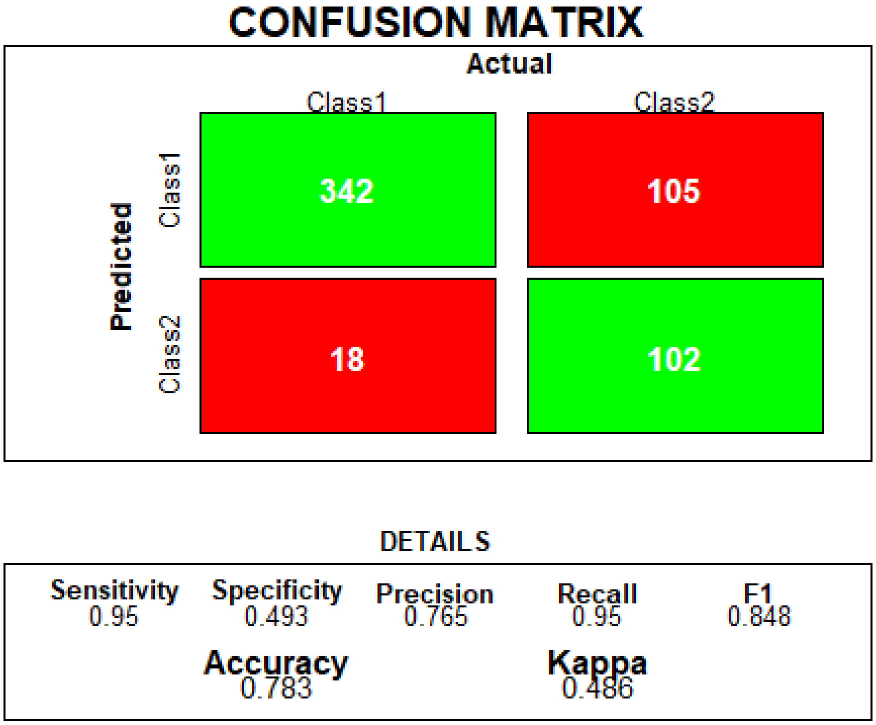
```
# perform cross-validation
svm.tune.L.out <- tune(
  svm,                   # SVM function
  TravelInsurance~.,                 # Formula for the model
  data = train_data,            # my training data frame
  kernel = "linear",     # Linear kernel
  ranges = list(cost = seq(0.01, 10, length.out = 20), scale=T)
svm.tune.L.out
```

Tried different values cost: seq(0.01, 10, length.out = 20), and we found that among the given sequence of cost, our model will be best when we use cost=0.01

With this cost we fit the best model for linear kernel

```r
# Svm model with liner kernel and cost=0.01

svm.L.best.mod=svm(TravelInsurance~., data = train_data, kernel = "linear", c
ost =0.01, scale = T )
```

We predicted our test data set with this model and found the result as:



CONFUSION MATRIX

		Actual	
		Class1	Class2
Predicted	Class1	342	105
	Class2	18	102

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.95	0.493	0.765	0.95	0.848

Accuracy	Kappa
0.783	0.486

B) Radial kernel

As discussed earlier for the radial kernel we can tune the hyperparameters both cost and gamma. We tuned them using cross-validation.

Tried different values of gamma: gamma=c(0.5,1,2,3,4) and cost: cost = seq(0.01, 10, length.out = 20)

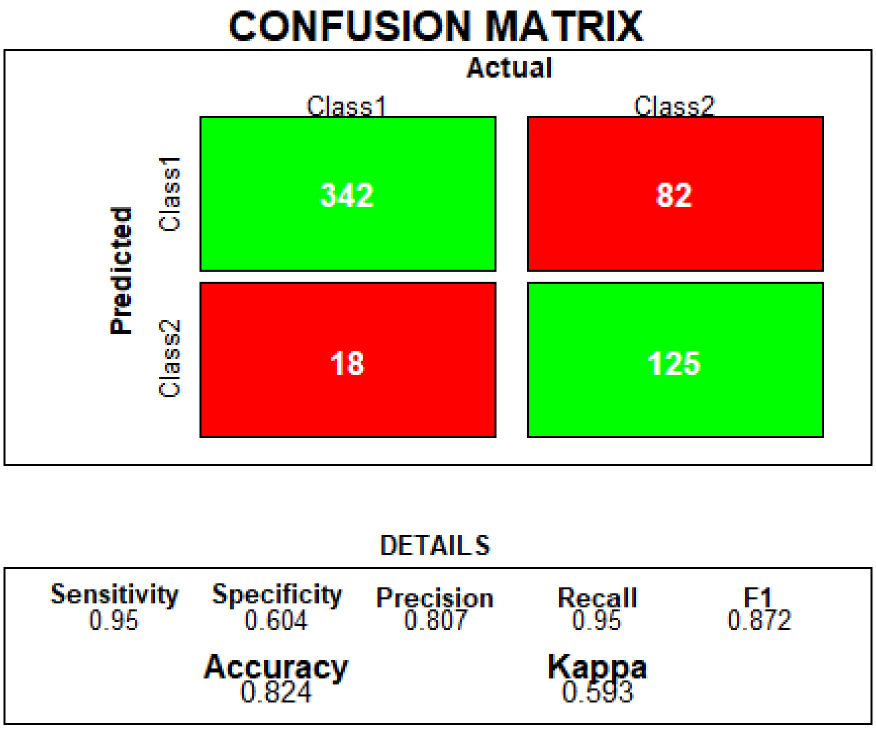
```r
# SVM with radial kernel

set.seed(100)
# perform cross-validation
svm.tune.R.out <- tune(
  svm,                     # SVM function
  TravelInsurance~.,                   # Formula for the model
  data = train_data,             # my training data frame
  kernel = "radial",       # Linear kernel
  ranges = list(cost = seq(0.01, 10, length.out = 20),gamma=c(0.5,1,2,3,4), s
cale=T)
```

```
svm.tune.R.out
```

We found that among the given sequence of cost and gamma , our model will be best when we use cost=1.061579 and gamma=0.05

```
# SVM with radial kernel with cost=1.061579 and gamma=0.5

svm.R.best.mod=svm(TravelInsurance~., data = train_data, kernel = "radial", c
ost =1.061579, gamma=0.5, scale = T )
summary(svm.R.best.mod)
```

predicting our test data set with this model, we obtain the following results.



CONFUSION MATRIX

Predicted \ Actual	Class1	Class2
Class1	342	82
Class2	18	125

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.95	0.604	0.807	0.95	0.872

Accuracy	Kappa
0.824	0.593

C) Polynomial kernel

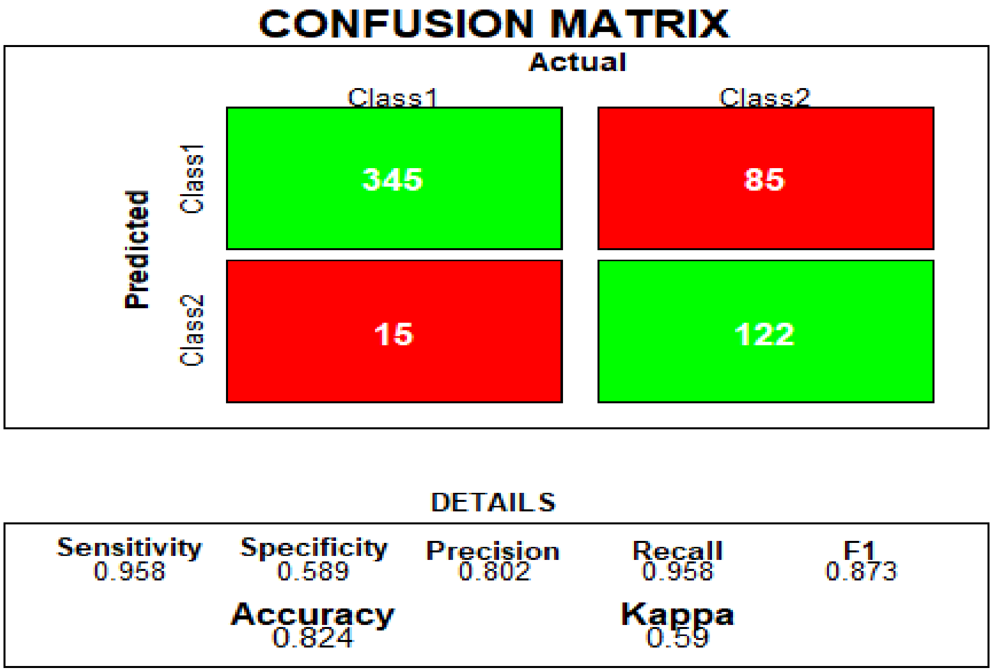
As discussed earlier for the polynomial kernel we can tune the hyperparameters both cost and degree. We tuned them using cross-validation.

```
set.seed(100)
# perform cross-validation
svm.tune.P.out <- tune(
  svm,                        # SVM function
  TravelInsurance~.,                      # Formula for the model
  data = train_data,                # my training data frame
  kernel = "polynomial",      # Linear kernel
  ranges = list(cost=seq(0.01, 10, length.out = 20),degree=c(0.25,0.33,0.5,1,
2,3,4), scale=T)

)
svm.tune.P.out
```

so, we will get best model with polynomial kernel when cost=50 and degree=4. We fit a model using this cost and degree.

```
# SVM with polynomial kernel, cost=50 and degree=4

svm.P.best.mod=svm(TravelInsurance~., data = train_data, kernel = "polynomial
", cost =6.845263, degree=4, scale = T )
summary(svm.P.best.mod)
```

When we predict our test data set with this model, we get the following results.



Model Comparison

All the model discussed above are classified below in the tabular form with their accuracy, precision and Recall.

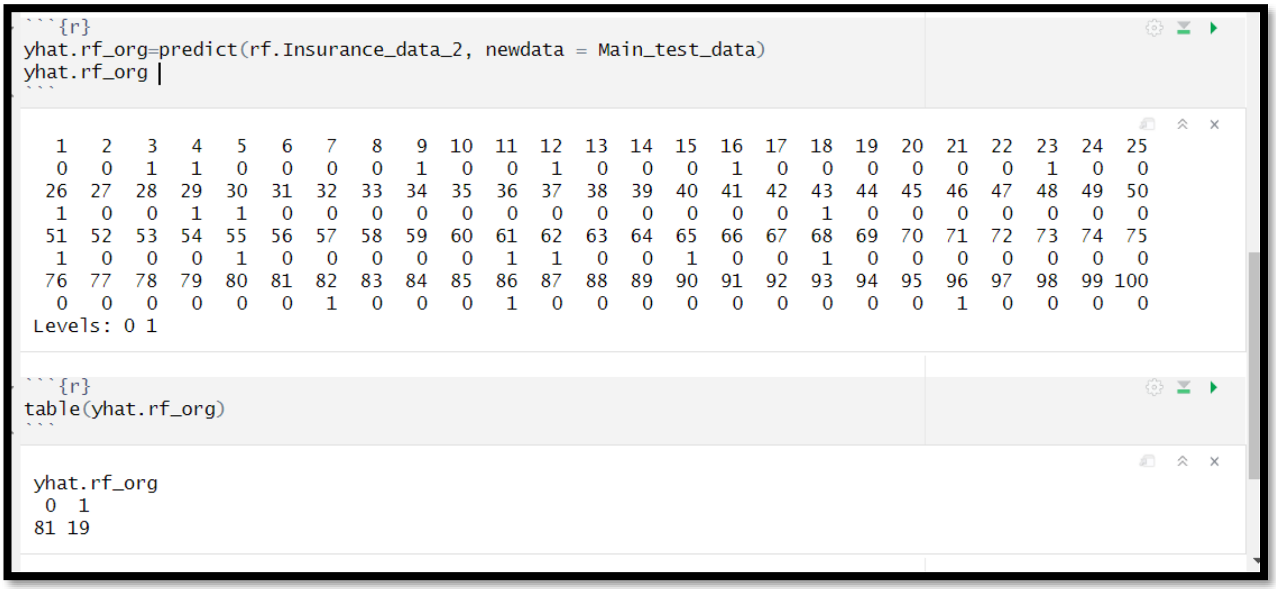
Machine Learning Models	**Accuracy**	**Precision**	**Recall**

Logistic Regression	78.5%	77.4%	93.3%
Linear Discriminant Analysis (LDA)	77.4%	76.5%	93.1%
Quadratic Discriminant Analysis (QDA)	76.7%	78.2%	87.8%
KNN- Classifier	78.1%	78.1%	91.1%
Decision Tee (Unpruned)	79.4%	81.6%	88.8%
Decision Tee (Pruned)	84.3%	82%	97.6%
Bagging	76.9%	81.6%	84%
Random Forest	**84.8%**	**82.7%**	**97.3%**
Boosting	81.8%	80.4%	94.4%
Support Vector Machine (Linear kernel)	78.3%	76.5%	95%
Support Vector Machine (Radial kernel)	82.4%	80.7%	95%
Support Vector Machine (Polynomial kernel)	82.4%	80.2%	87.3%

As highlighted earlier, choosing the optimal predictive model can involve various approaches. In simplifying our selection process, this paper considers the model's accuracy, precision, and recall. After a comprehensive comparison of all models, the analysis points towards Random Forest as the most effective choice for predicting our test dataset. Consequently, the final prediction for our test data will be executed utilizing the Random Forest model.

Prediction by Model

Finally, we made the prediction on the withheld 100 observations and found out that: among them 81 are not going to purchase the travel insurance and 19 are going to purchase the travel insurance. This is the prediction made by the Random Forest model. For individual prediction you can check the figure given below:



~~~
```{r}
yhat.rf_org=predict(rf.Insurance_data_2, newdata = Main_test_data)
yhat.rf_org
```
~~~

```
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 
  0   0   1   1   0   0   0   0   1   0   0   1   0   0   0   1   0   0   0   0   0   0   1   0   0 
 26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 
  1   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0 
 51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75 
  1   0   0   0   1   0   0   0   0   0   1   1   0   0   1   0   0   1   0   0   0   0   0   0   0 
 76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 
  0   0   0   0   0   0   1   0   0   0   1   0   0   0   0   0   0   0   0   0   1   0   0   0   0 
Levels: 0 1
```

~~~
```{r}
table(yhat.rf_org)
```
~~~

```
yhat.rf_org
 0  1 
81 19 
```

Recommendation

If the tour and travel company is planning to make a advertisement campaign in any form of social media, they should keep in mind these following point to increase the number of customers purchasing the Travel insurance

- Pay attention to customers who travel frequently abroad.
- Focus marketing efforts on individuals over 30 with chronic health conditions.
- Take note of the trend that wealthier customers show a higher interest in purchasing travel insurance.
- Enhance customer attraction by offering rewards and special promotions. And more…

References

- Confusion matrix: krlmlr, Zurich, Switzerland- https://stackoverflow.com/questions/23891140/r-how-to-visualize-confusion-matrix-using-the-caret-package