



# **INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE**

## **Predictive Analysis for Travel Insurance through Machine Learning**

PG-DBDA March 2024

Submitted By:  
Group No: 24

Roll No.	Name
243521	Heena There
243542	Renuka Rathod

**Mr. Abhijit Nagargoje**  
Project Guide

**Mr. Rohit Puranik**  
Centre Coordinator

## **ABSTRACT**

Travel insurance serves as a crucial financial safeguard, offering coverage against unforeseen expenses and losses incurred during travel. With the advent of the proliferation of insurance types and the amplified demand for Covid-related coverage, insurance companies face the imperative task of accurately predicting customers' likelihood to purchase insurance. This can assist the insurance providers in focusing on the most lucrative clients and boosting sales. By employing advanced machine learning techniques, this study aims to forecast the consumer segments most inclined to acquire travel insurance, allowing targeted strategies to be developed. A comprehensive analysis was carried out on a Kaggle dataset comprising prior clients of a travel insurance firm utilizing the K-Nearest Neighbours (KNN), Decision Tree Classifier (DT), Support Vector Machines (SVM), Naïve Bayes (NB), Logistic Regression (LR), and Random Forest (RF) models. Extensive data cleaning was done before model building. Performance evaluation was then based on accuracy, F1 score. Inexplicably, BernoulliNB() and GaussianNB() outperformed other models, achieving an accuracy of 0.71. The findings of this study are a valuable guide for deploying machine learning algorithms in predicting travel insurance purchases, thus empowering insurance companies to target the most lucrative client and bolster revenue generation.

### **ACKNOWLEDGEMENT**

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, Mr. Abhijit Nagargoje for providing me with the right guidance and advice at the crucial juncture and for showing me the right way. I extend my sincere thanks to our respected Centre Co-Ordinator Mr. Rohit Puranik, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Heena There (240341225021)

Renuka Rathod (240341225041)

**Table of Contents**

<b>Sr No</b>	<b>Description</b>	<b>Page No.</b>
1	Introduction	1
2	Software and Hardware requirements	3
3	Data Collection	4
4	Data Preprocessing	5
5	Illuminating Patterns and Trends	18
6	Model Selection	19
7	Model Comparison	24
8	Conclusion	25
9	References	26

## 1. INTRODUCTION

Travel insurance is a vital aspect of travel planning that provides travellers with financial protection against unforeseen events that may occur during a trip. These events could include medical emergencies, flight cancellations, lost or stolen luggage, and other travel-related mishaps. With the rise in global travel, the demand for travel insurance has grown exponentially, with travellers looking for the most cost-effective and comprehensive coverage.

This predictive model will utilize machine learning, data mining, and statistical analysis techniques to identify patterns and trends in the data. The model will examine historical data on past travel insurance purchases, demographic information, and travel itinerary to predict the likelihood of an individual purchasing travel insurance. In this project, we have some database history of the customer as a dataset. The target variable of this dataset is the customer will buy travel insurance or not. The goal of this project is to create a predictive model that can accurately predict the likelihood of an individual purchasing travel insurance based on various factors like Age, Income, Number of Family members etc.

➤ Purpose

Predicting travel insurance needs helps travelers choose the right coverage for their trips. It involves anticipating risks like medical emergencies or trip cancellations to select suitable insurance. This can prevent overpaying for unnecessary coverage or being underinsured. Accurate predictions make the process easier and ensure travelers are better protected and informed.

➤ Scope

The scope of predicting travel insurance needs typically includes analyzing various risk factors like travel destinations, trip duration, and planned activities. It involves developing models or tools to forecast potential risks and recommend appropriate coverage options. This project would also encompass gathering data, designing user-friendly interfaces, and ensuring the predictions are accurate and useful for travelers in making informed insurance choices.

➤ Objective of Project on Predictive Analysis for Travel Insurance:

The objective of the project on Predictive Analysis for Travel Insurance is to develop a robust analytical framework that can forecast travel-related risks and optimize insurance coverage. The primary goal is to leverage historical data and predictive modeling techniques to accurately estimate the likelihood of various travel issues, such as medical emergencies, trip cancellations, or lost baggage. By doing so, the project aims to provide personalized insurance recommendations based on individual travel profiles and risk factors.

The project involves collecting and preprocessing data related to travel behaviors, insurance claims, and risk events. Using advanced machine learning algorithms and statistical methods, the system will analyze patterns and trends to predict potential risks associated with different types of travel. This predictive analysis will help travelers select appropriate insurance plans that provide adequate coverage without unnecessary expense.

## 2. SYSTEM AND SOFTWARE REQUIREMENTS

### ➤ **Hardware Specifications:**

Machine: Desktop/Laptop

Operating System: Windows 10 (or above) or Linux 18 LTS(or above)

Processors: Intel core i5 (minimum)

Memory: 8 GB RAM or above

Hard Disk (SSD): 250GB or more

Video Card (optional): Intel Integrated Graphics (suggested – 4GB graphics card – NVIDIA)

Network: Ethernet / Wi-Fi with 25Mbps Speed Connection (UL/DL)

### ➤ **Software Specifications:**

Language: Python 3.7 or above (stable build)

Platform: Anaconda Latest stable build

Notebook: Google Colab

Libraries: Pandas, Numpy, Matplotlib, Seaborn, Plotly, Scikit Learn.

### 3. DATA COLLECTION

We have taken the data from Toffee Insurance Company, which consists of ten columns, for this research.

#### **Descriptive Statistics:**

Number of Rows-101001

Number of Columns-10

#### **Key Input Variables:**

- ✓ Age - The customer's age
- ✓ Employment Type - The Industry in which the customer works
- ✓ Graduate or Not - This refers to the customer's status as a college graduate
- ✓ Annual Income - The customer's annual income expressed in Indian rupees
- ✓ Family Member -The customer's Family size
- ✓ Chronic Diseases-To know if the customers have any serious medical conditions
- ✓ Frequent Flyer -To know how many customers are a frequent traveler
- ✓ Ever Travelled Abroad -To know if the customers ever traveled overseas
- ✓ Travel Insurance - To know if the customers bought the Travel Insurance

#### **Outcome variable:**

Binary Classification task determining whether the customer possesses travel insurance. (0 and 1)



## 4. DATA PREPROCESSING

In the first step, we have used below mentioned Libraries to read the data and by using file\_path data loading has been done:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
sns.set_style('whitegrid')
import warnings
warnings.filterwarnings('ignore')
```

### Step 1: Load the data

```
[145] file_path = '/content/TravelInsurancePrediction_Dataset.csv'
data = pd.read_csv(file_path)
```

To check the data types of the columns data.info() command given:

### Step 2: Understand the data

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101000 entries, 0 to 100999
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Age                    101000 non-null float64
1   Employment Type        101000 non-null object
2   GraduateOrNot           101000 non-null object
3   AnnualIncome            101000 non-null float64
4   FamilyMembers           101000 non-null float64
5   ChronicDiseases         101000 non-null float64
6   FrequentFlyer           101000 non-null object
7   EverTravelledAbroad     101000 non-null object
8   TravelInsurance         101000 non-null float64
dtypes: float64(5), object(4)
memory usage: 6.9+ MB
```

Converted the Float column into Integer:

```
# Convert all float columns to integer
float_columns = data.select_dtypes(include=['float64']).columns
data[float_columns] = data[float_columns].astype(int)

# Check the data types of the columns after conversion
print("\nData types after converting all float columns to integers:")
print(data.dtypes)
```



```
Data types after converting all float columns to integers:
Age                                int64
Employment Type                    object
GraduateOrNot                      object
AnnualIncome                       int64
FamilyMembers                      int64
ChronicDiseases                    int64
FrequentFlyer                      object
EverTravelledAbroad                object
TravelInsurance                    int64
dtype: object
```

```
[ ] data_types = data.dtypes
print("\nData Types:\n", data.dtypes)
```



```
Data Types:
Age                                int64
Employment Type                    object
GraduateOrNot                      object
AnnualIncome                       int64
```


After receiving the data, certain modifications were implemented to clean it up.

Handling Missing Values: The dataset underwent filtration to identify discrepancies or missing values. No gaps or missing data were identified.

Removing Column: An unnamed column was identified and removed during subsequent machine learning analysis, and its indexing for dashboard presentation was adjusted to start at 1 instead of 0.


Converting data: Additionally, four columns, namely (Employment Type, Graduate or Not, Frequent Flyer and Ever Travelled Abroad) indicating categorical data, were converted to numerical format by using one hot encoding method to enhance clarity and improve model performance.

```
[ ] missing_values = data.isnull().sum()
    print("\nMissing Values:\n", data.isnull().sum())
```



```
Missing Values:
Age                0
Employment Type    0
GraduateOrNot      0
AnnualIncome       0
FamilyMembers      0
ChronicDiseases    0
FrequentFlyer      0
EverTravelledAbroad 0
TravelInsurance    0
dtype: int64
```

```
data_summary = data.describe()
print("\nData Summary:\n", data.describe())
```



```
Data Summary:
```

	Age	AnnualIncome	FamilyMembers	ChronicDiseases	\
count	101000.000000	1.010000e+05	101000.000000	101000.000000	
mean	29.262604	9.328520e+05	4.393891	0.170069	
std	2.954213	3.608964e+05	1.828206	0.375696	
min	24.000000	2.999990e+05	1.000000	0.000000	
25%	27.000000	6.000000e+05	3.000000	0.000000	
50%	29.000000	9.000000e+05	4.000000	0.000000	
75%	32.000000	1.200000e+06	6.000000	0.000000	
max	35.000000	1.800000e+06	9.000000	1.000000	

After Preprocessing, please find few rows of the dataset:

First 5 Rows of Data:

	Age	Employment Type	GraduateOrNot	AnnualIncome
0	30	Government Sector	Yes	899999
1	25	Government Sector	Yes	1300000
2	33	Government Sector	Yes	1300000
3	33	Government Sector	Yes	399999
4	32	Private Sector/Self Employed	Yes	350000

	FamilyMembers	ChronicDiseases	FrequentFlyer	EverTravelledAbroad
0	3	0	No	No
1	5	0	Yes	Yes
2	4	1	No	No
3	6	0	No	No
4	2	0	No	No

	TravelInsurance
0	0
1	0
2	0
3	0
4	0

Separated the Target variable and then Training and Testing Data:

```

v Separate X & Y

[ ] X = data.drop('TravelInsurance', axis=1)
    Y = data['TravelInsurance']

▶ X.shape,Y.shape
((37665, 8), (37665,))

[ ] Start coding or generate with AI.

v Train Test Split

[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=7, stratify=Y)


    X_train.shape, X_test.shape, Y_train.shape, Y_test.shape


((26365, 8), (11300, 8), (26365,), (11300,))

```

Categorical Column converted into Numerical column:

### One hot encoding


 `x_train.dtypes`



	0
<b>Age</b>	int64
<b>Employment Type</b>	object
<b>GraduateOrNot</b>	object
<b>AnnualIncome</b>	int64
<b>FamilyMembers</b>	int64
<b>ChronicDiseases</b>	int64
<b>FrequentFlyer</b>	object
<b>EverTravelledAbroad</b>	object

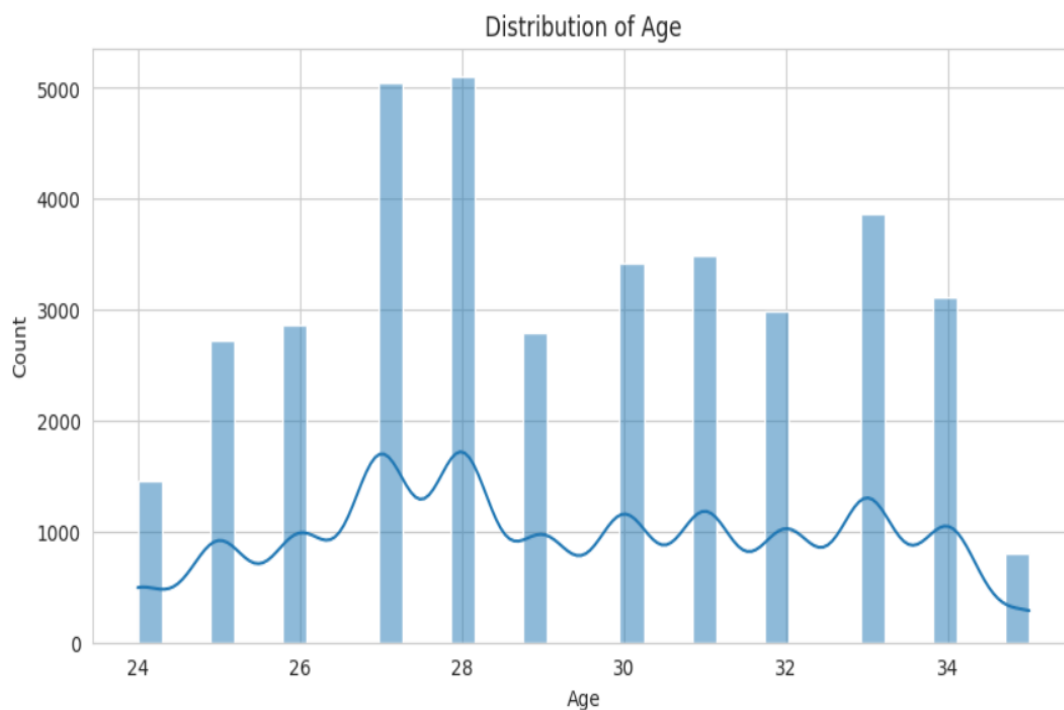
**dtype:** object

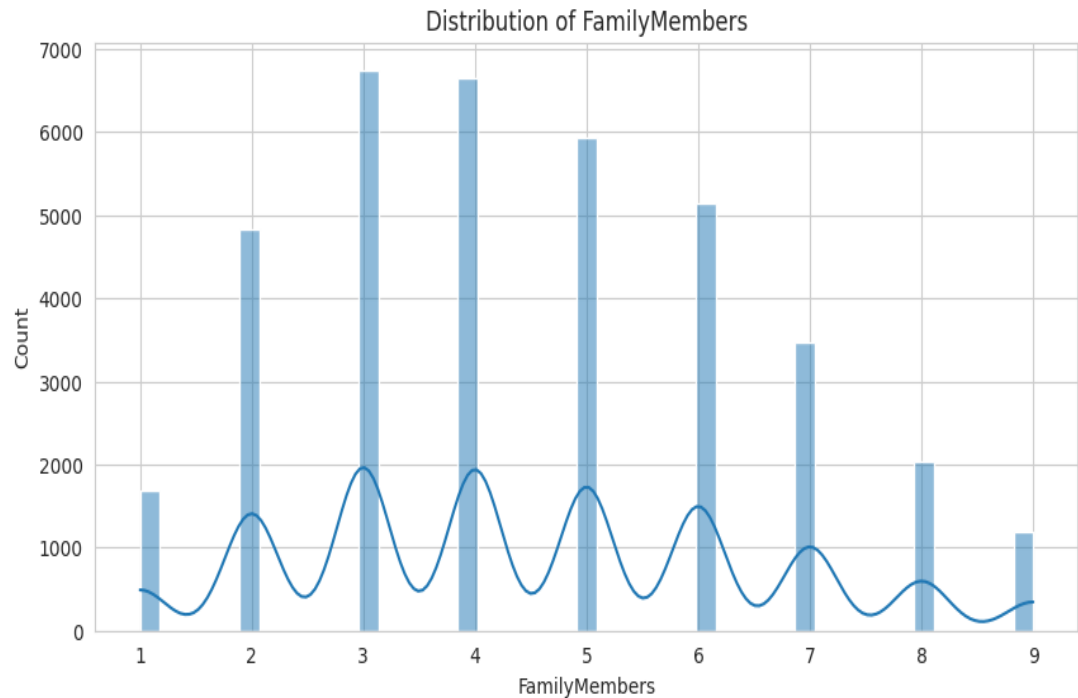
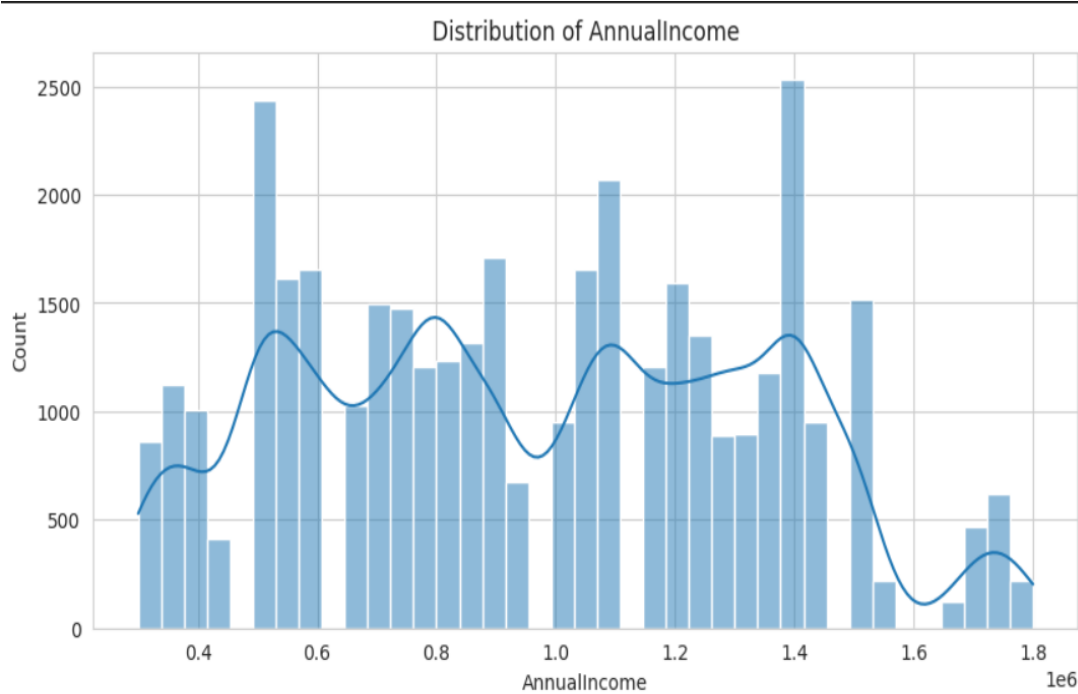
```
[ ] x_train_ohe = pd.get_dummies(x_train)
    x_train.shape, x_train_ohe.shape
```

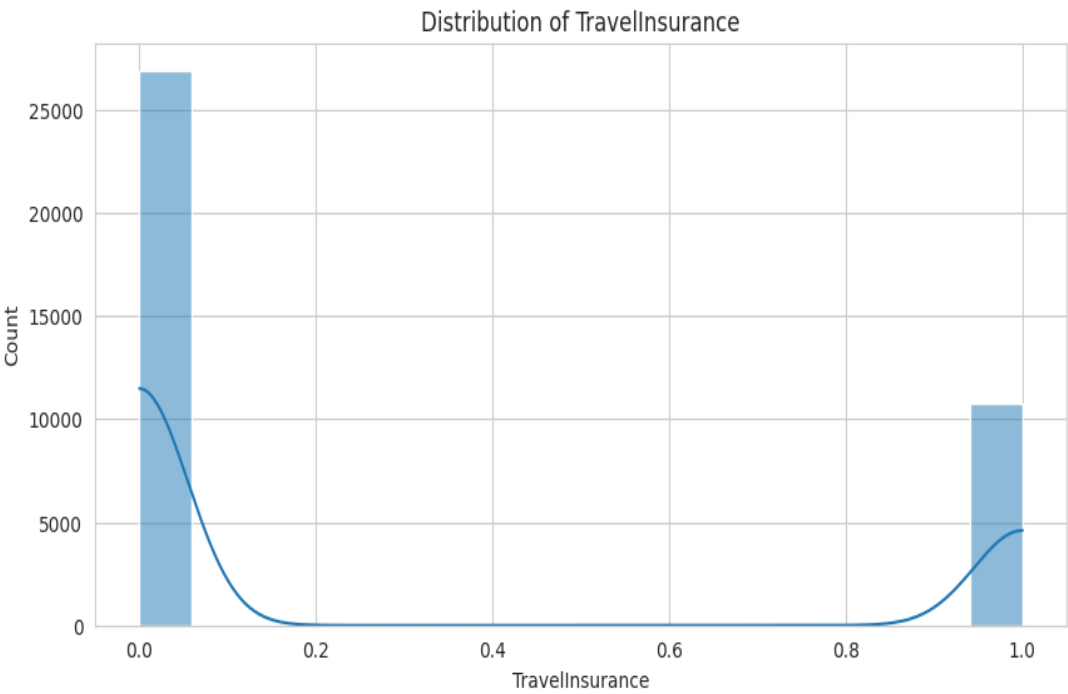
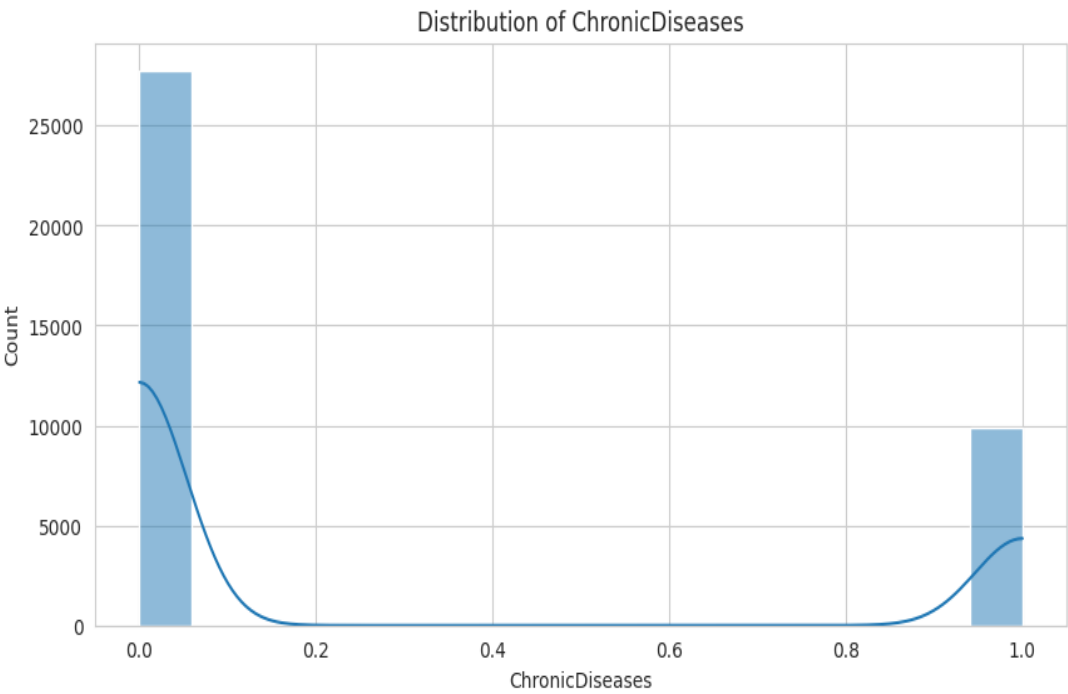
 `((26365, 8), (26365, 12))`

Univariate Analysis done for the distribution for numerical features as to understand the distribution and characteristics of that single variable without considering the influence of any other variables.

```
# Distribution of numerical features
num_features = data.select_dtypes(include=[np.number]).columns
for feature in num_features:
    plt.figure(figsize=(10, 5))
    sns.histplot(data[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()
```









To find the outliers following methods are used:

1.

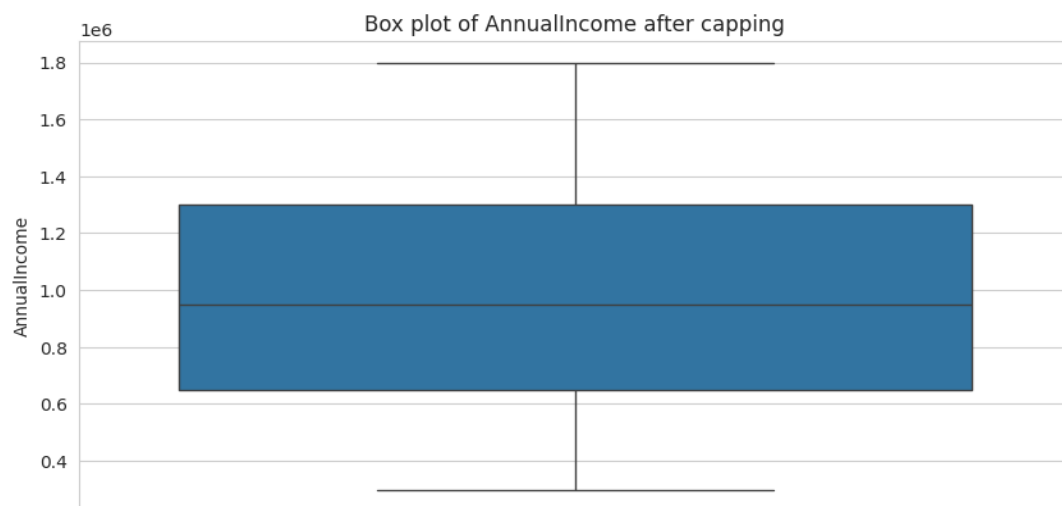
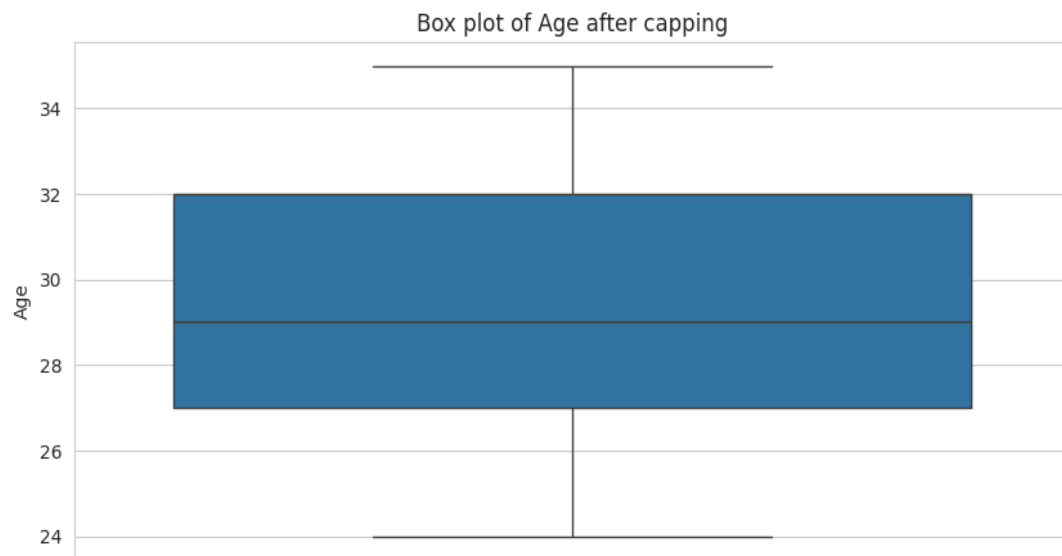
```
num_features = data.select_dtypes(include=[np.number]).columns
for feature in num_features:
    outliers = detect_outliers_iqr(data, feature)
    print(f'Outliers detected in {feature}: {len(outliers)}')
```

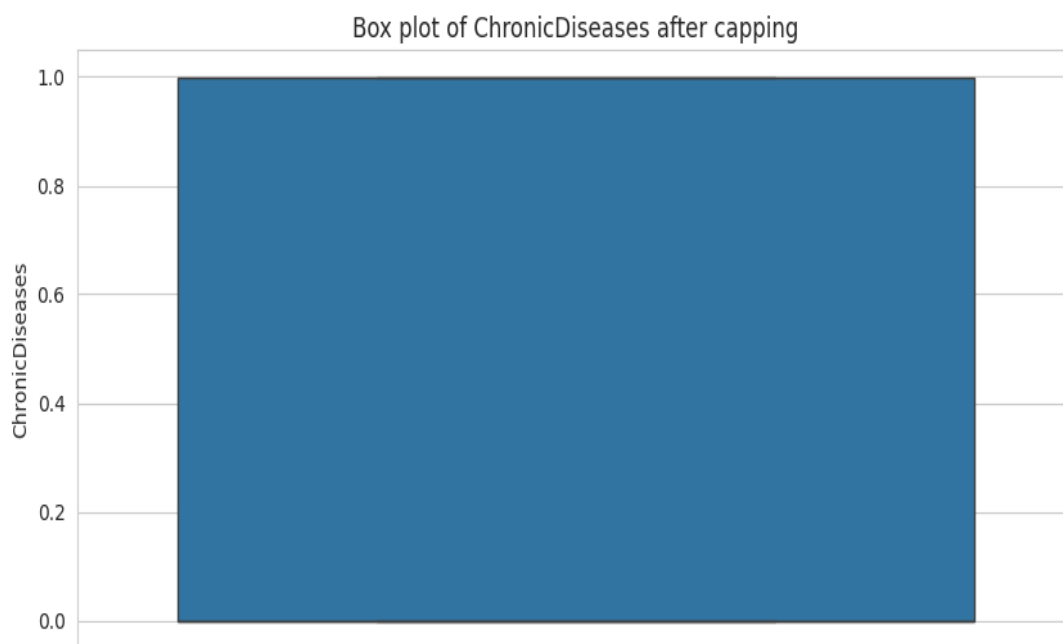
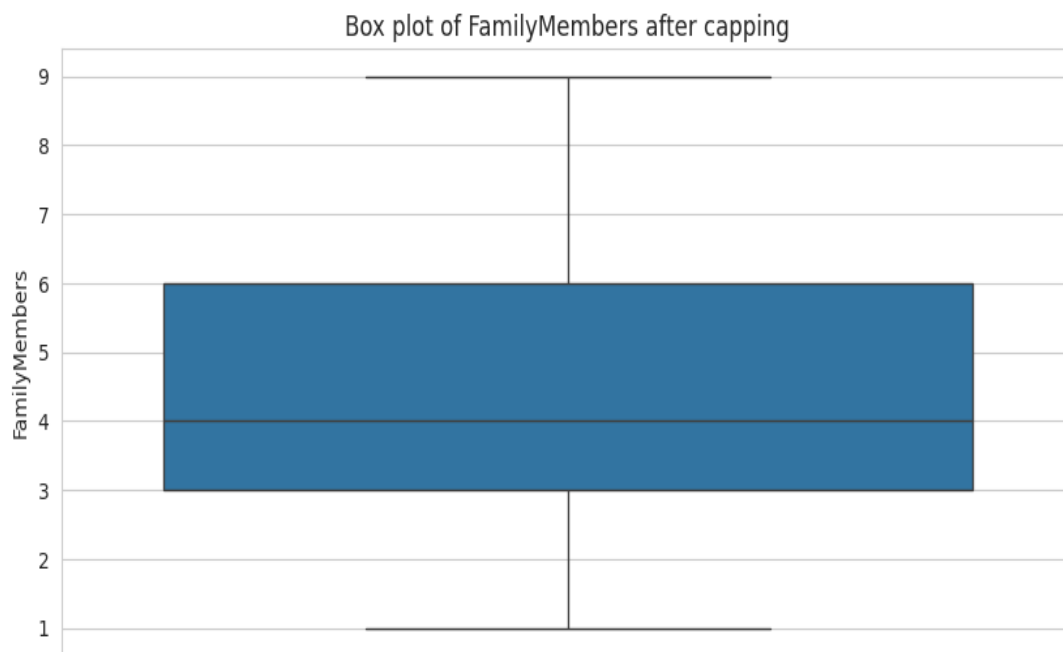
```
Outliers detected in Age: 0
Outliers detected in AnnualIncome: 0
Outliers detected in FamilyMembers: 0
Outliers detected in ChronicDiseases: 0
Outliers detected in TravelInsurance: 0
```

2.

```
def cap_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
    df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
    return df

for feature in num_features:
    data = cap_outliers(data, feature)
    # Check the box plot after capping
    plt.figure(figsize=(10, 5))
    sns.boxplot(data[feature])
    plt.title(f'Box plot of {feature} after capping')
    plt.show()
```





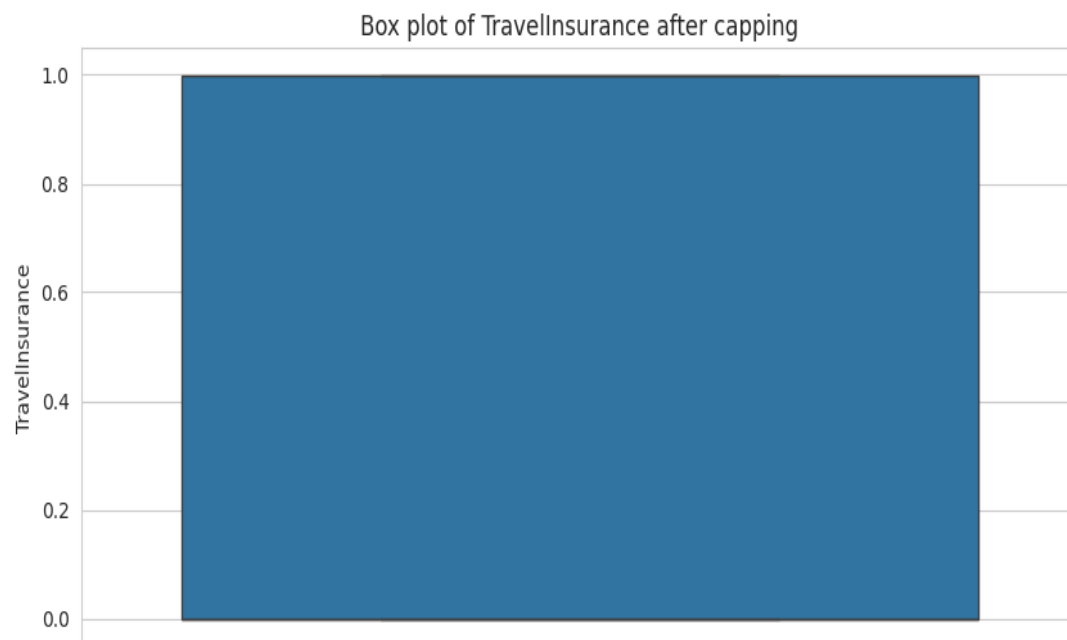
**3.**

```
[ ] # Calculate the z-scores for each numerical column
z_scores = np.abs((data.select_dtypes(include=np.number) -
                  data.select_dtypes(include=np.number).mean()) / data.select_dtypes(include=np.number).std())

# Identify any values with z-score greater than 3
potential_outliers = data[(z_scores > 3).any(axis=1)]

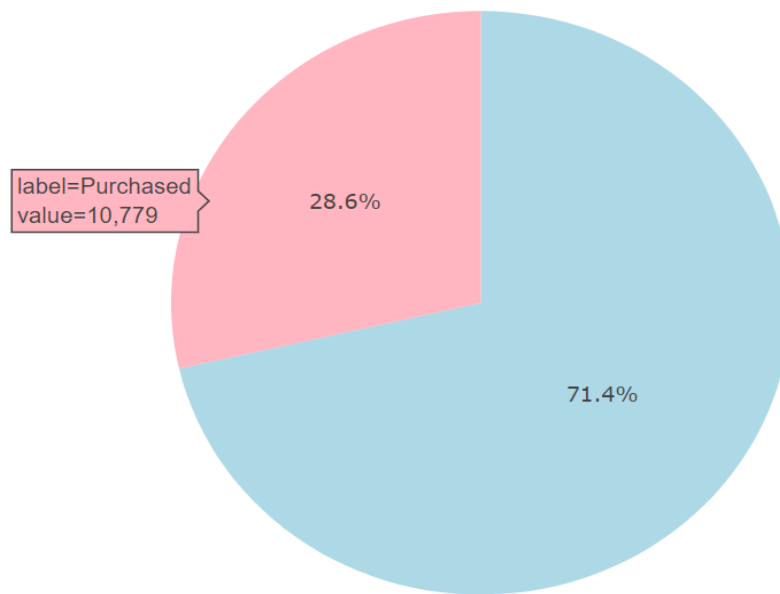
# Print the potential outliers
print(potential_outliers)
```

↳ Empty DataFrame  
Columns: [Age, Employment Type, GraduateOrNot, AnnualIncome, FamilyMembers, ChronicDiseases, FrequentFlyer, EverTravelledAbroad, TravelInsurance]  
Index: []



## 5. ILLUMINATING PATTERNS AND TRENDS

Visualization in machine learning not only aids in understanding data and model behavior but facilitates effective communication of results to stakeholders. It helps make informed decisions, troubleshoot issues, and build trust in machine learning models.



Out of the company's 37,652 customers, only 28.6% opted to purchase the travel insurance package and 71.4% of people have not bought the travel insurance.

## 6. MODEL SELECTION

The process of model selection involves choosing appropriate machine learning algorithms that are well-suited to the problem at hand—in this case, predicting the likelihood of customers purchasing travel insurance. Several algorithms were considered based on their suitability for classification tasks, their ability to handle different types of data, and their performance characteristics. Below are the models that were considered:

- **Logistic Regression:** This is a simple yet effective algorithm for binary classification problems. It models the probability of the default class (in this case, whether a customer will purchase travel insurance) using a logistic function. Logistic Regression was considered because it provides interpretable results and performs well when the relationship between features and the target variable is approximately linear.
- **Decision Trees:** Decision Trees are a versatile algorithm that splits the data into subsets based on the values of input features, creating a tree-like structure. They are easy to interpret and can handle both numerical and categorical data. Decision Trees were chosen because they provide a clear understanding of feature importance and are capable of capturing non-linear relationships between features and the target variable.
- **Random Forest:** Random Forest is an ensemble method that builds multiple Decision Trees and merges them to produce a more robust and accurate prediction. It reduces the risk of overfitting and improves generalization by averaging the results from multiple trees. Random Forest was considered because it tends to perform well on a wide range of problems and can handle large datasets with high-dimensional features.
- **Support Vector Machines (SVM):** SVMs are powerful classifiers that work by finding the hyperplane that best separates the data into classes. SVMs are effective in high-dimensional spaces and are useful when the number of dimensions exceeds the number of samples. They were considered for their ability to handle complex, non-linear relationships.

- Naive Bayes: is a probabilistic classifier that uses Bayes' Theorem with the assumption that features are independent given the class label. It calculates the likelihood of data belonging to each class and assigns the class with the highest probability. It's simple, fast, and works well for text classification.
- k-Nearest Neighbours (k-NN): is a non-parametric classifier that assigns a data point to the class most common among its k nearest neighbours in the feature space. It's intuitive and adaptable but can be computationally expensive and sensitive to the choice of k-NN and distance metric.

### **Training Process:**

- The data was first preprocessed to ensure it was in the correct format for training. This involved encoding categorical variables, normalizing numerical features, and handling missing values.
- The model was then trained on the training set, learning the relationship between the features and the target variable.
- Cross-validation was employed to further validate the model's performance and ensure that it was not overfitting to the training data. This technique involves splitting the training set into several smaller sets, training the model on each set, and averaging the results.

### **Evaluation Metrics:**

Several evaluation metrics were used to assess the performance of the model:

- Accuracy: The proportion of correctly predicted instances out of the total instances. It is a useful metric when the classes are balanced, but less informative when there is a class imbalance.
- Precision: The ratio of true positive predictions to the total number of positive predictions. It indicates how many of the predicted positive cases were actually positive.

- Recall (Sensitivity): The ratio of true positive predictions to the total number of actual positive cases. It measures how well the model identifies true positive cases.
- F1-Score: The harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, especially useful when dealing with imbalanced classes.

The model's performance was evaluated based on these metrics, and the results were compared across different models to select the best-performing one.

```
✓ Classifications

[ ] from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.svm import SVC
    from sklearn.linear_model import LogisticRegression
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
```



- **Logistics Regression:**

```
lr = LogisticRegression(class_weight='balanced',random_state = 7,solver = 'saga')
lr.fit(X_train_ohc,Y_train)
lr.coef_
```

```
array([[ -1.20514974e-06,  1.07585978e-07, -1.36718978e-07,
        -3.02234022e-08, -3.09326117e-08, -1.26412280e-08,
        -3.52342333e-08, -8.33960639e-09, -1.34820775e-08,
        -3.00917621e-08, -1.15844720e-08, -3.19893677e-08]])
```

```
[ ] Y_pred = lr.predict(X_test_ohc)
Y_test.shape,Y_pred.shape
from sklearn.metrics import classification_report
print(classification_report(Y_test,Y_pred))
```

```
precision    recall  f1-score   support

      0       0.00      0.00      0.00      8066
      1       0.29      1.00      0.45      3234

 accuracy          0.29      11300
 macro avg       0.14      0.50      0.22      11300
 weighted avg    0.08      0.29      0.13      11300
```

- **Support Vector Machine Algorithm:**

```
svc1 = SVC(class_weight='balanced', random_state=7)
svc1.fit(X_train_ohc, Y_train)
Y_pred_svc = svc1.predict(X_test_ohc)
print("SVC Classification Report:\n",classification_report(Y_test,Y_pred_svc))
```

```
SVC Classification Report:
precision    recall  f1-score   support

      0       0.76      0.70      0.73      8066
      1       0.37      0.43      0.40      3234

 accuracy          0.63      11300
 macro avg       0.56      0.57      0.56      11300
 weighted avg    0.64      0.63      0.63      11300
```

- **MultinomialNB() Algorithm**

```
[ ] mnb = MultinomialNB()
    mnb.fit(X_train_ohe,Y_train)
    Y_pred_mnb = mnb.predict(X_test_ohe)
    print("MultinomialNB Classification Report:\n",classification_report(Y_test,Y_pred_mnb))
```

```
↔ MultinomialNB Classification Report:
      precision    recall  f1-score   support

     0       0.76      0.72      0.74      8066
     1       0.38      0.43      0.40      3234

 accuracy      0.64      11300
 macro avg     0.57      0.57      0.57      11300
 weighted avg   0.65      0.64      0.64      11300
```

- **KNeighbors Classifier Algorithm**

```
[ ] knn = KNeighborsClassifier()
    knn.fit(X_train_ohe,Y_train)
    Y_pred_knn = knn.predict(X_test_ohe)
    print("KNeighborsClassifier Classification Report:\n",classification_report(Y_test,Y_pred_knn))
```

```
↔ KNeighborsClassifier Classification Report:
      precision    recall  f1-score   support

     0       0.73      0.80      0.76      8066
     1       0.33      0.24      0.28      3234

 accuracy      0.64      11300
 macro avg     0.53      0.52      0.52      11300
 weighted avg   0.61      0.64      0.62      11300
```

- **RandomForestClassifier Algorithm:**

```
rfc = RandomForestClassifier(max_samples= 0.8,oob_score = True,random_state=7)
rfc.fit(X_train_ohe,Y_train)
Y_pred_rfc = rfc.predict(X_test_ohe)
print("RandomForestClassifier Classification Report:\n",classification_report(Y_test,Y_pred_rfc))
```

```
RandomForestClassifier Classification Report:
              precision    recall  f1-score   support

      0       0.68       0.72       0.70       8066
      1       0.17       0.14       0.16       3234

 accuracy          0.56       11300
 macro avg       0.42       0.43       0.43       11300
 weighted avg    0.53       0.56       0.54       11300
```

- **DecisionTree Classifier Algorithm:**

```
dtc = DecisionTreeClassifier(criterion='gini', random_state=7,class_weight= 'balanced')
dtc.fit(X_train_ohe,Y_train)
Y_pred_dtc = dtc.predict(X_test_ohe)
print("DecisionTreeClassifier Classification Report:\n",classification_report(Y_test,Y_pred_dtc))
```

```
DecisionTreeClassifier Classification Report:
              precision    recall  f1-score   support

      0       0.67       0.60       0.63       8066
      1       0.21       0.27       0.23       3234

 accuracy          0.50       11300
 macro avg       0.44       0.43       0.43       11300
 weighted avg    0.54       0.50       0.52       11300
```

## 7. MODEL COMPARISON

As per Observation, BernoulliNB and GaussianNB algorithm gives the best accuracy 71%.

Sr. No.	Models	Accuracy
1	BernoulliNB()	71%
2	GaussianNB()	71%
3	K-Nearest Neighbors (KNN)	65%
4	MultinomialNB()	65%
5	Support Vector Machines (SVM)	63%
6	Random Forest Classifier	57%
7	DecisionTreeClassifier	52%
8	LogisticRegression	29%

## 8. CONCLUSION

In conclusion, this project has successfully developed a predictive model for travel insurance purchases, offering valuable insights into customer behaviour and actionable recommendations for the business. While the model performs well, there is always room for improvement and further exploration. By continuing to refine the model, enrich the data, and explore new avenues for personalization and real-time predictions, the business can stay ahead in a competitive market and better meet the needs of its customers.

We applied six different classification models, including Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbour (KNN), Naive Bayes (NB) and Support Vector Machines (SVC) to classify the data and determined that the Naive Bayes (NB) algorithm provided the best accuracy score of 71%, making it the recommended model for further use.

## 9.REFERENCES

- [https://en.wikipedia.org/wiki/Machine\\_learning](https://en.wikipedia.org/wiki/Machine_learning)
- [https://scikit-learn.org/stable/modules/linear\\_model.html#logistic-regression](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
- <https://scikit-learn.org/stable/modules/svm.html>
- <https://scikit-learn.org/stable/modules/neighbors.html>
- [https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)
- <https://scikit-learn.org/stable/modules/tree.html>
- <https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles>
- <https://www.kaggle.com/code/cindynz/travel-insurance-prediction-capstone-project#Conclusion>: