A third-party travel insurance servicing company that is based in Singapore.

The attributes:

1. Target: Claim Status (Claim.Status)
2. Name of agency (Agency)
3. Type of travel insurance agencies (Agency.Type)
4. Distribution channel of travel insurance agencies (Distribution.Channel)
5. Name of the travel insurance products (Product.Name)
6. Duration of travel (Duration)
7. Destination of travel (Destination)
8. Amount of sales of travel insurance policies (Net.Sales)
9. Commission received for travel insurance agency (Commission)
10. Gender of insured (Gender)
11. Age of insured (Age)

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.utils import resample
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn import tree
from scipy.stats import skew
from sklearn import preprocessing

import warnings
warnings.filterwarnings('ignore')
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
```

```python
# EDA Processing
# Importing Dataset
df = pd.read_csv("project_ml_travel_insurance.csv")
df.head()
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender
0	3433	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	0	7	MALAYSIA	0.0	17.82	NaN
1	4339	EPX	Travel Agency	Online	Cancellation Plan	0	85	SINGAPORE	69.0	0.00	NaN
2	34590	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	0	11	MALAYSIA	19.8	11.88	NaN
3	55816	EPX	Travel Agency	Online	2 way Comprehensive Plan	0	16	INDONESIA	20.0	0.00	NaN
4	13816	EPX	Travel Agency	Online	Cancellation Plan	0	10	KOREA, REPUBLIC OF	15.0	0.00	NaN

```python
# Identification of data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50553 entries, 0 to 50552
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   ID                    50553 non-null  int64  
 1   Agency                50553 non-null  object 
 2   Agency Type           50553 non-null  object 
 3   Distribution Channel  50553 non-null  object 
 4   Product Name          50553 non-null  object 
 5   Claim                 50553 non-null  int64  
 6   Duration              50553 non-null  int64  
 7   Destination           50553 non-null  object 
 8   Net Sales             50553 non-null  float64
 9   Commision (in value)  50553 non-null  float64
 10  Gender                14600 non-null  object 
 11  Age                   50553 non-null  int64  
dtypes: float64(2), int64(4), object(6)
memory usage: 4.6+ MB
```

```python
# Statistical Summary of Numeric Variables
df.describe()
```

	ID	Claim	Duration	Net Sales	Commision (in value)	Age
count	50553.000000	50553.000000	50553.000000	50553.000000	50553.00000	50553.000000
mean	31679.740134	0.014658	49.425969	40.800977	9.83809	40.011236
std	18288.265350	0.120180	101.434647	48.899683	19.91004	14.076566
min	0.000000	0.000000	-2.000000	-389.000000	0.00000	0.000000
25%	15891.000000	0.000000	9.000000	18.000000	0.00000	35.000000
50%	31657.000000	0.000000	22.000000	26.500000	0.00000	36.000000
75%	47547.000000	0.000000	53.000000	48.000000	11.55000	44.000000
max	63325.000000	1.000000	4881.000000	810.000000	283.50000	118.000000

```python
# Non-Graphical Univariate Analysis
print(df['Agency'].value_counts())
df['Agency'].unique()
print(df['Agency Type'].value_counts())
df['Agency Type'].unique()
print(df['Distribution Channel'].value_counts())
df['Distribution Channel'].unique()
print(df['Product Name'].value_counts())
df['Product Name'].unique()
print(df['Destination'].value_counts())
df['Destination'].unique()
print(df['Gender'].value_counts())
df['Gender'].unique()
```

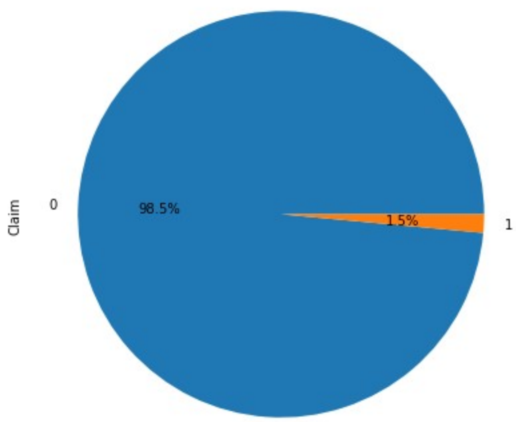
```
EPX    28002
CWT     6840
C2B     6631
JZI     5059
SSI      839
JWT      606
RAB      571
LWC      548
TST      421
KML      317
ART      272
CCR      158
CBH       81
TTW       77
CSR       68
ADM       63
```

```
Name: Agency, dtype: int64
Travel Agency    36575
Airlines         13978
Name: Agency Type, dtype: int64
Online     49665
Offline      888
Name: Distribution Channel, dtype: int64
Cancellation Plan                       14872
2 way Comprehensive Plan                10482
Rental Vehicle Excess Insurance          6840
Basic Plan                               4376
Bronze Plan                              3246
1 way Comprehensive Plan                 2648
Value Plan                               2169
Silver Plan                              1789
Annual Silver Plan                       1156
Ticket Protector                          839
Travel Cruise Protect                     421
Comprehensive Plan                        293
Gold Plan                                 292
24 Protect                                199
Single Trip Travel Protect Gold           159
Premier Plan                              158
Annual Gold Plan                          148
Single Trip Travel Protect Silver         133
Annual Travel Protect Gold                 81
Annual Travel Protect Silver               73
Individual Comprehensive Plan              58
Single Trip Travel Protect Platinum        57
Annual Travel Protect Platinum             45
Spouse or Parents Comprehensive Plan       12
Child Comprehensive Plan                    7
Name: Product Name, dtype: int64
SINGAPORE                   10608
MALAYSIA                     4747
THAILAND                     4699
CHINA                        3836
AUSTRALIA                    2934
                            ...
ZAMBIA                          2
BHUTAN                          1
CAYMAN ISLANDS                  1
TURKMENISTAN                    1
NORTHERN MARIANA ISLANDS        1
Name: Destination, Length: 102, dtype: int64
M    7527
F    7073
Name: Gender, dtype: int64
```

```
array([nan, 'F', 'M'], dtype=object)
```

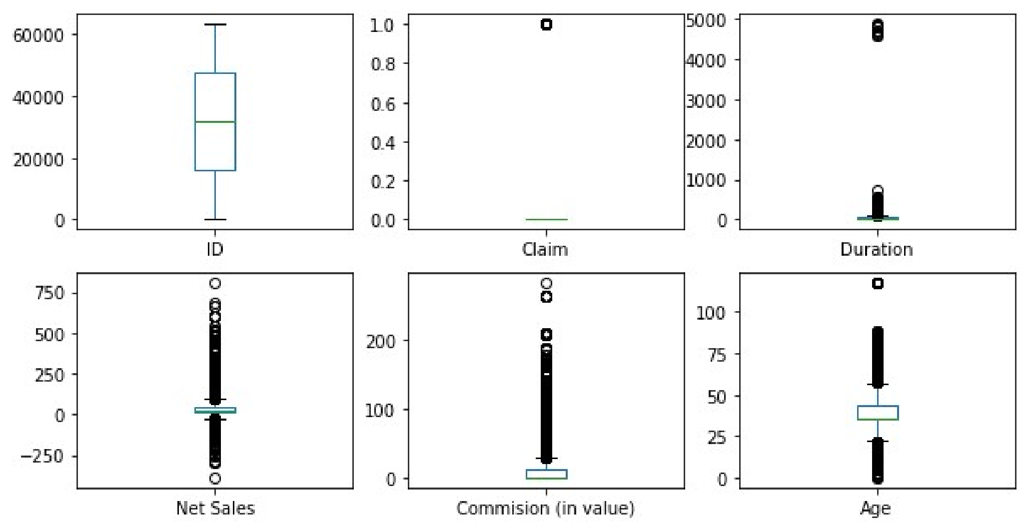
In [6]:
```
print(df["Claim"].value_counts())
print("--------------------------------------------------")
plt.figure(figsize=(7,7))
df["Claim"].value_counts().plot.pie(autopct="%.1f%%")
plt.show()
```

```
0    49812
1      741
Name: Claim, dtype: int64
--------------------------------------------------
```

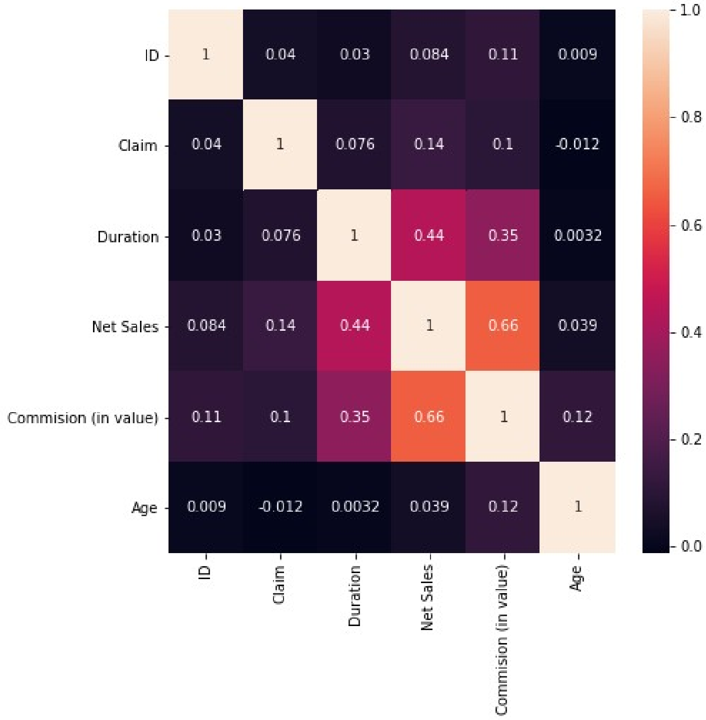



```
# the target variable is imbalanced . % of no claim is [49812*100/50553] = 98.5
df.plot(kind= 'box' , subplots=True, layout=(3,3), sharex=False, sharey=False, figsize=(10,8))
```

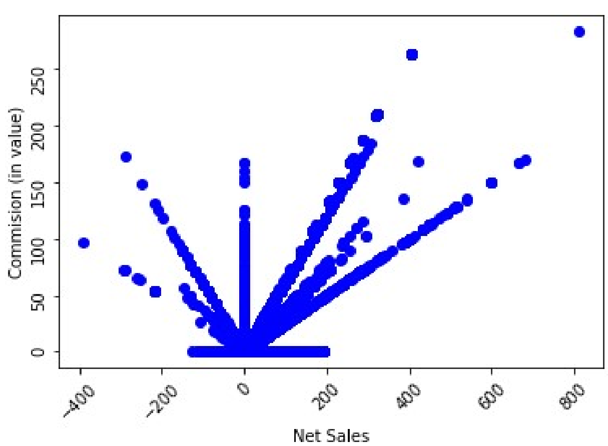
```
ID                      AxesSubplot(0.125,0.657941;0.227941x0.222059)
Claim                 AxesSubplot(0.398529,0.657941;0.227941x0.222059)
Duration              AxesSubplot(0.672059,0.657941;0.227941x0.222059)
Net Sales               AxesSubplot(0.125,0.391471;0.227941x0.222059)
Commision (in value)  AxesSubplot(0.398529,0.391471;0.227941x0.222059)
Age                   AxesSubplot(0.672059,0.391471;0.227941x0.222059)
dtype: object
```



```python
# Bivariate analysis
# heat map

corr = df.corr()
plt.figure(figsize=(7,7))
sns.heatmap(corr,annot=True)
plt.show()
```

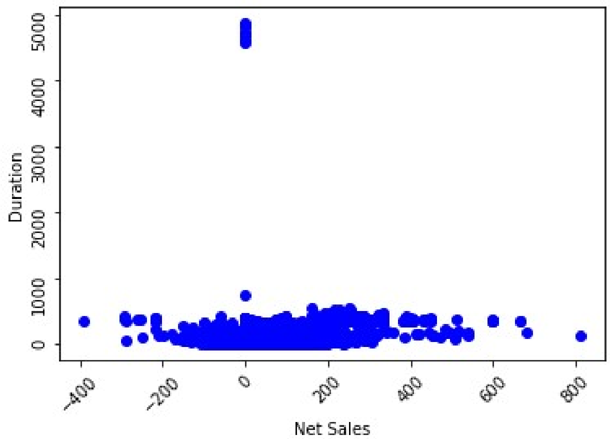



```python
# We see that there is correlation between Commision and Net Sales ie 0.66
# We see that there is correlation between Duration and Net Sales ie 0.44
# Very less correlation between Claim and numerical variables
# scatter plot
plt.figure()

plt.scatter(df["Net Sales"], df["Commision (in value)"], color = "Blue")
plt.xlabel("Net Sales")
plt.ylabel("Commision (in value)")
plt.xticks(rotation=45)
plt.yticks(rotation=90)
plt.show()
```

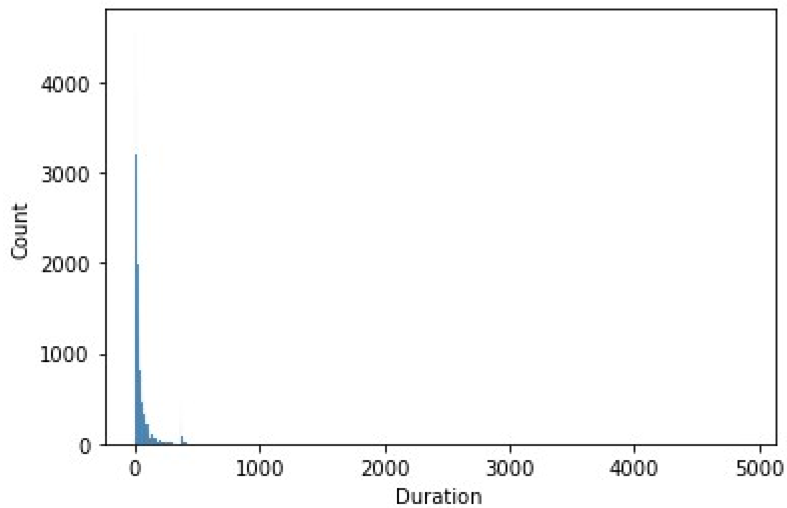


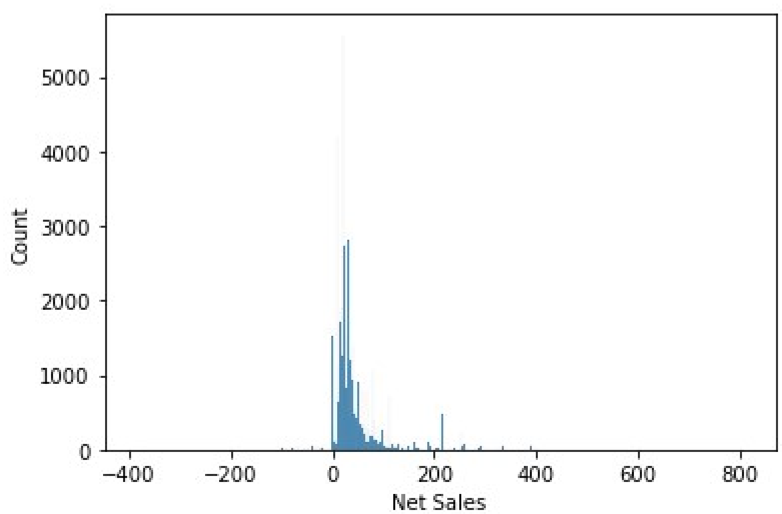
Net Sales

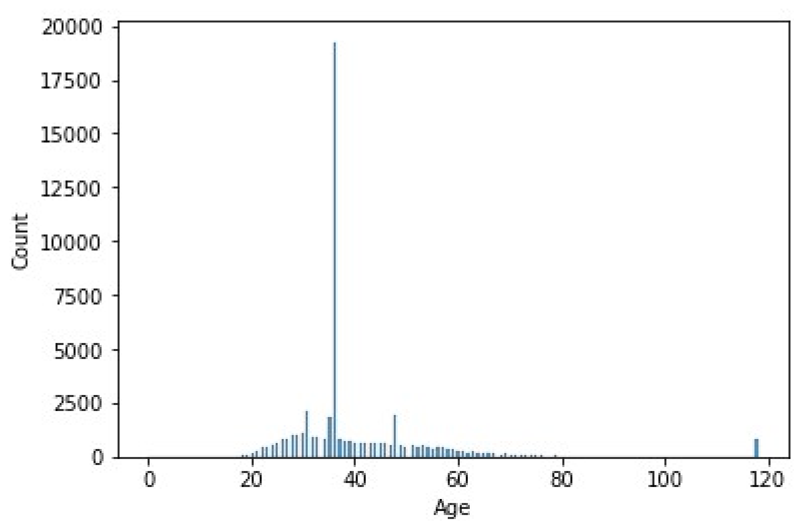
```python
# Some linear relationship is there between Net Sales and Commission
plt.figure()

plt.scatter(df["Net Sales"], df["Duration"], color = "Blue")
plt.xlabel("Net Sales")
plt.ylabel("Duration")
plt.xticks(rotation=45)
plt.yticks(rotation=90)
plt.show()
```




```python
# No linear relationship between Duration and Net sales
# Plot of numerical variable to study skewness
from scipy.stats import skew
num_col = ["Duration","Net Sales","Age"]
df_num = df[num_col]
for col in df_num:
    plt.figure()
    sns.histplot(x = df_num[col])
plt.show()
for col in df_num:
    print( col , skew(df_num[col]))
```








```
Duration 22.872063891229274
Net Sales 3.3281441910342053
Age 2.9783898494112435
```

```
# Since skew values are very high the outliers are removed
# Handling Outliers
# For "Age" attribute, we have very high values of up to 118.
# For "Duration" attribute, we have very high values of up to 4881, and also negative values.
# We assume that duration is measured in days.
# Hence negative values of duration are assumed to be error
# For "Commission (in value)", the highest value of 283.5 seems unreasonable.
# For "Net Sales", we observe a large range of values for this field with minimum -389.
# There is possibility of negative values due to cross-subsidies.

# Hence we remove rows where
# Age > 110
# Commission (in value) > 250
# Duration > 4000
```

```
df[(df["Age"]>110)]
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender	Age
90	41201	JWT	Airlines	Online	Value Plan	0	58	INDIA	78.0	31.20	F	118
108	31332	JWT	Airlines	Online	Value Plan	0	15	INDIA	31.0	12.40	M	118
140	40733	JWT	Airlines	Online	Value Plan	0	8	INDIA	39.0	15.60	M	118
153	5275	JWT	Airlines	Online	Value Plan	0	4	INDIA	78.0	31.20	F	118
181	46888	JWT	Airlines	Online	Value Plan	0	0	INDIA	31.0	12.40	M	118
...	...	...	...	...	...	...	...	...	...	...	...	...
50158	23267	JWT	Airlines	Online	Value Plan	0	41	INDIA	60.0	24.00	M	118
50179	13909	JWT	Airlines	Online	Value Plan	0	62	INDIA	31.0	12.40	M	118
50250	3401	JWT	Airlines	Online	Value Plan	0	15	INDIA	31.0	12.40	M	118
50429	21940	JZI	Airlines	Online	Basic Plan	0	19	SRI LANKA	35.0	12.25	NaN	118
50478	14579	CCR	Travel Agency	Offline	Comprehensive Plan	0	6	THAILAND	29.0	9.57	F	118

795 rows × 12 columns

```
df.drop(df[df['Age'] >110].index, inplace = True)
```

```
df[(df["Commision (in value)"]>250)]
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender
5479	60736	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	371	UNITED STATES	404.25	262.76	F
10402	37706	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	385	UNITED STATES	404.00	262.60	F
14564	60564	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	365	UNITED STATES	404.25	262.76	M
16541	60737	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	369	UNITED STATES	404.25	262.76	F
22012	39631	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	366	UNITED STATES	404.00	262.60	F
24141	57703	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	378	UNITED STATES	404.25	262.76	M
29989	43334	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	364	UNITED STATES	404.25	262.76	M
37086	58504	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	365	UNITED STATES	404.25	262.76	M

```
df.drop(df[df['Age'] >110].index, inplace = True)
```

```
df[(df["Commision (in value)"]>250)]
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender	Age
5479	60736	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	371	UNITED STATES	404.25	262.76	F	38
10402	37706	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	385	UNITED STATES	404.00	262.60	F	30
14564	60564	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	365	UNITED STATES	404.25	262.76	M	42
16541	60737	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	369	UNITED STATES	404.25	262.76	F	51
22012	39631	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	366	UNITED STATES	404.00	262.60	F	53
24141	57703	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	378	UNITED STATES	404.25	262.76	M	45
29989	43334	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	364	UNITED STATES	404.25	262.76	M	44
37086	58504	LWC	Travel Agency	Online	Annual Travel Protect Platinum	0	365	UNITED STATES	404.25	262.76	M	58

```
df.drop(df[df["Commision (in value)"] >250].index, inplace = True)
```

```
df[(df["Duration"]>4000)]
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender	Age
12083	49722	SSI	Airlines	Online	Ticket Protector	0	4609	SINGAPORE	0.32	0.09	NaN	48
15326	55326	SSI	Airlines	Online	Ticket Protector	0	4580	SINGAPORE	0.32	0.09	NaN	48
16797	39842	SSI	Airlines	Online	Ticket Protector	0	4685	SINGAPORE	0.32	0.09	NaN	48
17702	30826	SSI	Airlines	Online	Ticket Protector	0	4736	SINGAPORE	0.32	0.09	NaN	48
23844	9232	SSI	Airlines	Online	Ticket Protector	0	4844	SINGAPORE	0.32	0.09	NaN	48
27270	6847	SSI	Airlines	Online	Ticket Protector	0	4857	SINGAPORE	0.32	0.09	NaN	48
28143	15281	SSI	Airlines	Online	Ticket Protector	0	4815	SINGAPORE	0.32	0.09	NaN	48
30465	41391	SSI	Airlines	Online	Ticket Protector	0	4652	SINGAPORE	0.32	0.09	NaN	48
31070	12438	SSI	Airlines	Online	Ticket Protector	0	4829	SINGAPORE	0.32	0.09	NaN	48
42137	30437	SSI	Airlines	Online	Ticket Protector	0	4738	SINGAPORE	0.32	0.09	NaN	48
43742	3025	SSI	Airlines	Online	Ticket Protector	0	4881	SINGAPORE	0.13	0.04	NaN	48

```
df.drop(df[df["Duration"] >4000].index, inplace = True)
```

```
df[(df["Duration"]>4000)]
```

	ID	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Commision (in value)	Gender	Age
12083	49722	SSI	Airlines	Online	Ticket Protector	0	4609	SINGAPORE	0.32	0.09	NaN	48
15326	55326	SSI	Airlines	Online	Ticket Protector	0	4580	SINGAPORE	0.32	0.09	NaN	48
16797	39842	SSI	Airlines	Online	Ticket Protector	0	4685	SINGAPORE	0.32	0.09	NaN	48
17702	30826	SSI	Airlines	Online	Ticket Protector	0	4736	SINGAPORE	0.32	0.09	NaN	48
23844	9232	SSI	Airlines	Online	Ticket Protector	0	4844	SINGAPORE	0.32	0.09	NaN	48
27270	6847	SSI	Airlines	Online	Ticket Protector	0	4857	SINGAPORE	0.32	0.09	NaN	48
28143	15281	SSI	Airlines	Online	Ticket Protector	0	4815	SINGAPORE	0.32	0.09	NaN	48
30465	41391	SSI	Airlines	Online	Ticket Protector	0	4652	SINGAPORE	0.32	0.09	NaN	48
31070	12438	SSI	Airlines	Online	Ticket Protector	0	4829	SINGAPORE	0.32	0.09	NaN	48
42137	30437	SSI	Airlines	Online	Ticket Protector	0	4738	SINGAPORE	0.32	0.09	NaN	48
43742	3025	SSI	Airlines	Online	Ticket Protector	0	4881	SINGAPORE	0.13	0.04	NaN	48

```
# After removing outliers
df.info()
```

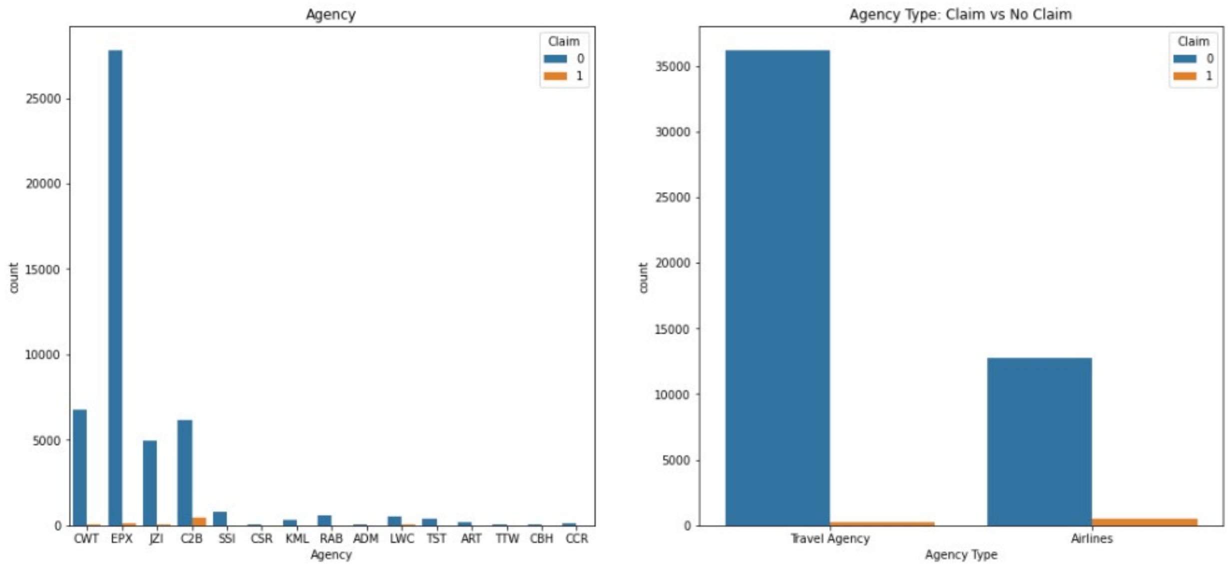
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49739 entries, 0 to 50552
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   ID                    49739 non-null  int64  
 1   Agency                49739 non-null  object 
 2   Agency Type           49739 non-null  object 
 3   Distribution Channel  49739 non-null  object 
 4   Product Name          49739 non-null  object 
 5   Claim                 49739 non-null  int64  
 6   Duration              49739 non-null  int64  
 7   Destination           49739 non-null  object 
 8   Net Sales             49739 non-null  float64
 9   Commision (in value)  49739 non-null  float64
 10  Gender                13910 non-null  object 
 11  Age                   49739 non-null  int64  
dtypes: float64(2), int64(4), object(6)
memory usage: 4.9+ MB
```

```
# the dataset has reduced from 50553 rows to 49739 rows
# removing columns which do not contribute
# In gender there are only 13910 non null values
# Percentage of Non Null Values = [13910*100/49739] = 26.51
# Percentage of Null Values = 100 - 26.51 = 73.49
# Column ID is just Customer Number and does not contribute to the claim
# Hence Column ID and Gender are dropped from analysis
```

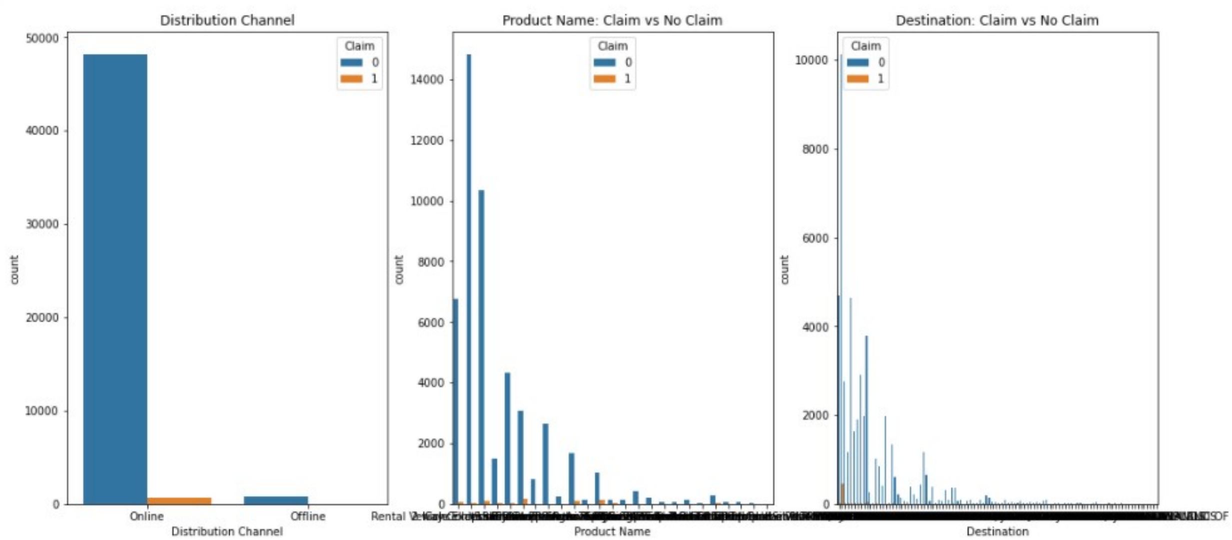
```
df.drop(["ID", "Gender"], axis = 1, inplace = True)
```

```python
# Studying Relationship of Categorical Variables with target variable
```

```python
fig, ax = plt.subplots(1, 2, figsize = (18, 8))
sns.countplot("Agency", hue = "Claim",data = df, ax= ax[0])
ax[0].set_title("Agency")
sns.countplot("Agency Type", hue = "Claim",data = df, ax= ax[1])
ax[1].set_title("Agency Type: Claim vs No Claim")
plt.show()
```




```python
fig, ax = plt.subplots(1, 3, figsize = (18, 8))
sns.countplot("Distribution Channel", hue = "Claim",data = df, ax= ax[0])
ax[0].set_title("Distribution Channel")
sns.countplot("Product Name", hue = "Claim",data = df, ax= ax[1])
ax[1].set_title("Product Name: Claim vs No Claim")
sns.countplot("Destination", hue = "Claim",data = df, ax= ax[2])
ax[2].set_title("Destination: Claim vs No Claim")
plt.show()
```



```
# Performing Label Encoding of Categorical Variables
cat_col = ["Agency","Agency Type","Distribution Channel","Product Name","Destination","Claim"]
num_col = ["Duration","Net Sales","Age"]
df_num = df[num_col]
df_cat = df[cat_col]
```

```
for col in df_cat:
    le = LabelEncoder()
    df_cat[col] = le.fit_transform(df_cat[col])
```

```
# Final Data Set with Numerical Values
df_new = pd.concat([df_num,df_cat],axis=1)

df_new.head()
```

	Duration	Net Sales	Age	Agency	Agency Type	Distribution Channel	Product Name	Destination	Claim
0	7	0.0	31	6	1	1	16	56	0
1	85	69.0	36	7	1	1	10	79	0
2	11	19.8	75	6	1	1	16	56	0
3	16	20.0	32	7	1	1	1	38	0
4	10	15.0	29	7	1	1	10	47	0

```
# Checking values of target variable Claim
print(df['Claim'].value_counts())
df['Claim'].unique()
```

```
0    49007
1      732
Name: Claim, dtype: int64
```

```
array([0, 1], dtype=int64)
```

```
# Since target variable has value 0 and 1 the problem is of classfication
```

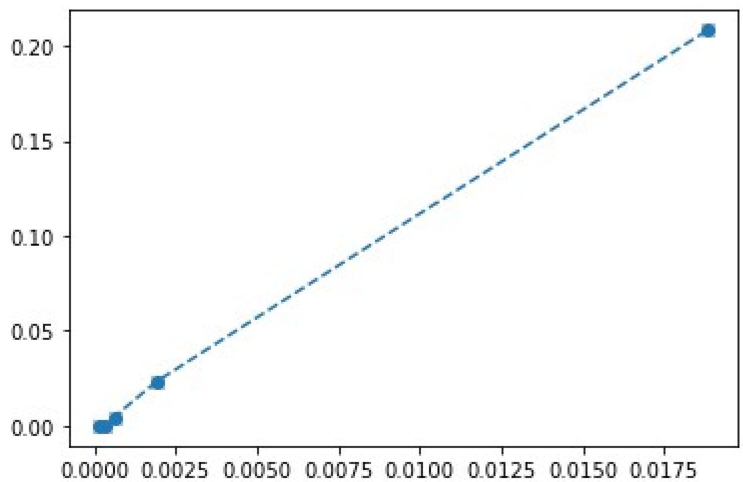
```
def create_model(model,X_train,y_train):
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test,y_pred))
    tn, fp, fn, tp = confusion_matrix(y_test,y_pred).ravel()
    print(tp,fp)
    print(fn,tn)
    roc= roc_auc_score(y_test,y_pred)
    print("ROC  AUC value = ", round(roc,3))
    return model
```

```
# Model 1 LogisticRegression
# Baseline Model
y = df_new.Claim
X = df_new.drop("Claim",axis=1)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
```

```
lr = LogisticRegression()
create_model(lr,X_train,y_train)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     14706
           1       0.00      0.00      0.00       216

    accuracy                           0.99     14922
   macro avg       0.49      0.50      0.50     14922
weighted avg       0.97      0.99      0.98     14922

0 2
216 14704
ROC  AUC value =  0.5

LogisticRegression()
```

```
# Plotting ROC AUC curve for different thresholds
prob = lr.predict_proba(X_test)[:,-1]
thresholds= [0.5,0.4,0.3,0.2,0.1]
tprs = []
fprs = []
for th in thresholds:
 y_pred = np.where(prob >= th, 1 , 0)
 tn, fp, fn, tp = confusion_matrix(y_test,y_pred).ravel()
 tpr = tp/(tp + fn)
 fpr = fp/(fp + tn)

 tprs.append(tpr)
 fprs.append(fpr)
plt.figure()
plt.plot(fprs,tprs,"x--")
plt.scatter(fprs,tprs)
plt.show()
```




```
# Model 2 RandomForest
rf=RandomForestClassifier(n_estimators=100)
create_model(rf,X_train,y_train)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     14706
           1       0.16      0.02      0.03       216

    accuracy                           0.98     14922
   macro avg       0.57      0.51      0.51     14922
weighted avg       0.97      0.98      0.98     14922

4 21
212 14685
ROC  AUC value =  0.509

RandomForestClassifier()
```

```python
# Since f1-score for 1 is 0 for Logistic Regression and 0.02 for RandomForrest
# we have to remove the imbalance in the dataset
```

```python
# Removing imbalance in training dataset by undersampling the training dataset
# Model 3
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=1)

y = df_new.Claim
X = df_new.drop("Claim",axis=1)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
X_sample1, y_sample1 = rus.fit_resample(X_train,y_train)
pd.Series(y_sample1).value_counts()
```

```
1    516
0    516
Name: Claim, dtype: int64
```

```python
lr2 = LogisticRegression()
create_model(lr2,X_sample1, y_sample1)
```

```
              precision    recall  f1-score   support

           0       0.99      0.81      0.90     14706
           1       0.05      0.69      0.10       216

    accuracy                           0.81     14922
   macro avg       0.52      0.75      0.50     14922
weighted avg       0.98      0.81      0.88     14922

148 2724
68 11982
ROC  AUC value =  0.75

LogisticRegression()
```

```python
# ROC AUC value has increased
# Removing imbalance in training dataset by oversampling the training dataset
# Model 4
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=1)

y = df_new.Claim
X = df_new.drop("Claim",axis=1)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
X_sample1, y_sample1 = ros.fit_resample(X_train,y_train)
pd.Series(y_sample1).value_counts()
```

```
1    34301
0    34301
Name: Claim, dtype: int64
```

```python
lr2 = LogisticRegression()
create_model(lr2,X_sample1, y_sample1)
```

```
              precision    recall  f1-score   support

           0       0.99      0.82      0.90     14706
           1       0.05      0.66      0.10       216

    accuracy                           0.82     14922
   macro avg       0.52      0.74      0.50     14922
weighted avg       0.98      0.82      0.89     14922

143 2596
73 12110
ROC  AUC value =  0.743

LogisticRegression()
```

```python
# Since undersampling  and oversampling results are close we use oversampling for further analysis
# as it has more rows
```

```python
# Ensembling techniques
# Model 5
rf=RandomForestClassifier(n_estimators=100)
create_model(rf,X_sample1, y_sample1)
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     14706
           1       0.10      0.05      0.07       216

    accuracy                           0.98     14922
   macro avg       0.54      0.52      0.53     14922
weighted avg       0.97      0.98      0.98     14922

11 103
205 14603
ROC  AUC value =  0.522

RandomForestClassifier()
```

```
# Model 6
ada = AdaBoostClassifier(n_estimators=100)
create_model(ada,X_sample1, y_sample1)
```

```
              precision    recall  f1-score   support

           0       0.99      0.80      0.89     14706
           1       0.05      0.69      0.09       216

    accuracy                           0.80     14922
   macro avg       0.52      0.75      0.49     14922
weighted avg       0.98      0.80      0.88     14922

149 2912
67 11794
ROC  AUC value =  0.746

AdaBoostClassifier(n_estimators=100)
```

```
# Model 7
from sklearn.svm import LinearSVC
svc = LinearSVC(random_state=1)
create_model(svc,X_sample1, y_sample1)
```

```
              precision    recall  f1-score   support

           0       0.99      0.88      0.93     14706
           1       0.06      0.50      0.10       216

    accuracy                           0.88     14922
   macro avg       0.53      0.69      0.52     14922
weighted avg       0.98      0.88      0.92     14922

109 1758
107 12948
ROC  AUC value =  0.693

LinearSVC(random_state=1)
```

```
# since f1 score of 1 has only slightly increased from 0.02 to 0.1 after resampling the training dataset
# the entire dataset is resampled
```

```
# Model 8
# upsampling of entire dataset by creating new dataset df_upsampled on which model of Logistic Regression
# algorithm will be trained
df_majority = df_new[df_new.Claim==0]
df_minority = df_new[df_new.Claim==1]
df_minority_upsampled = resample(df_minority,replace=True,n_samples=49007,random_state=1)
df_upsampled = pd.concat([df_majority,df_minority_upsampled])
df_upsampled.Claim.value_counts()
```

```
1    49007
0    49007
Name: Claim, dtype: int64
```

```
y = df_upsampled.Claim
X = df_upsampled.drop("Claim",axis=1)
```

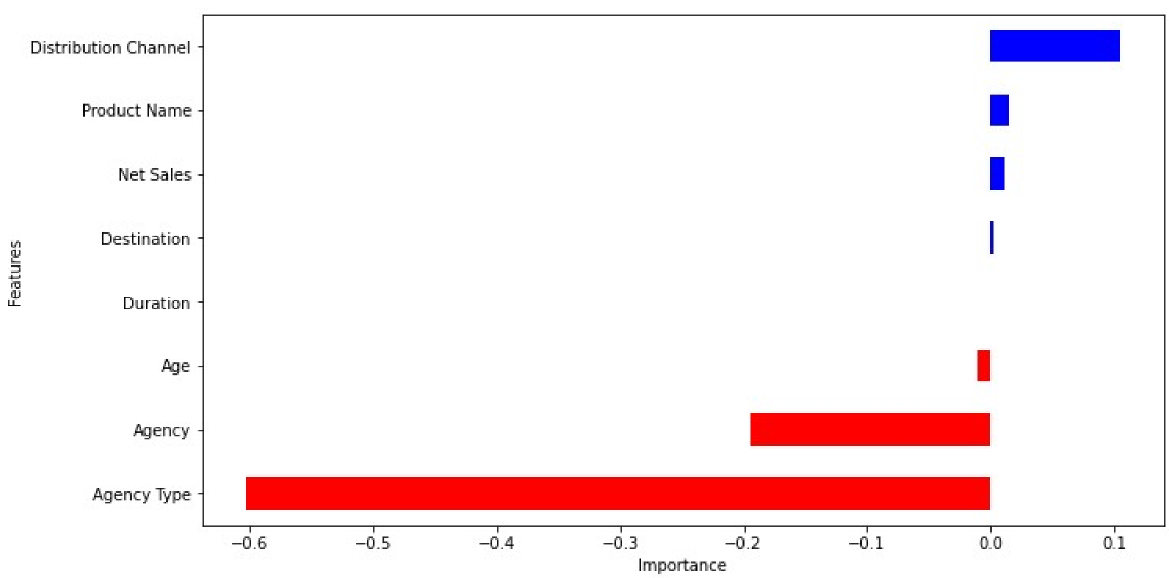
```
y = df_upsampled.Claim
X = df_upsampled.drop("Claim",axis=1)
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
lr = LogisticRegression()
create_model(lr,X_train, y_train)
```

```
              precision    recall  f1-score   support

           0       0.71      0.84      0.77     14756
           1       0.80      0.65      0.72     14649

    accuracy                           0.74     29405
   macro avg       0.75      0.74      0.74     29405
weighted avg       0.75      0.74      0.74     29405

9535 2423
5114 12333
ROC  AUC value =  0.743

LogisticRegression()
```

```
coeff = list(lr.coef_[0])
labels = list(X_train.columns)
features = pd.DataFrame()
features['Features'] = labels
features['importance'] = coeff
features.sort_values(by=['importance'], ascending=True, inplace=True)
features['positive'] = features['importance'] > 0
features.set_index('Features', inplace=True)
features.importance.plot(kind='barh', figsize=(11, 6),color = features.positive.map({True: 'blue', False: 'red'}))
plt.xlabel('Importance')
```

```
Text(0.5, 0, 'Importance')
```



```
# we see that destination and duration do not contribute significantly to the model
# Model 9 downsampling to create new dataset df_downsampled on which model
# of Logistic Regression will be trained

# Downsampling
df_majority = df_new[df_new.Claim==0]
df_minority = df_new[df_new.Claim==1]
df_majority_downsampled = resample(df_majority,replace=False,n_samples=732,random_state=1)
df_downsampled = pd.concat([df_majority_downsampled,df_minority])
df_downsampled.Claim.value_counts()
```

```
1    732
0    732
Name: Claim, dtype: int64
```

```
y = df_downsampled.Claim
X = df_downsampled.drop("Claim",axis=1)
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
lr = LogisticRegression()
create_model(lr,X_train,y_train)
```

```
              precision    recall  f1-score   support

           0       0.69      0.85      0.76       220
           1       0.80      0.62      0.70       220

    accuracy                           0.73       440
   macro avg       0.75      0.73      0.73       440
weighted avg       0.75      0.73      0.73       440

137 34
83 186
ROC  AUC value =  0.734

LogisticRegression()
```

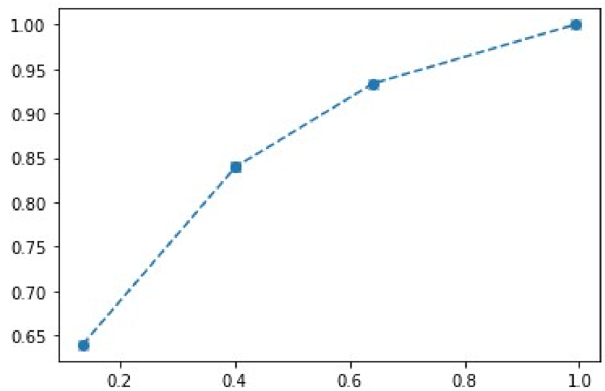
```
# Since upsampling gives better ROC  AUC value we use upsampled data set for further analysis
```

```
# Model 10 Decision Tree Algorithm with Pruning
y = df_upsampled.Claim
X = df_upsampled.drop("Claim",axis=1)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
```

```
clf = DecisionTreeClassifier(max_depth = 3,random_state = 1)
create_model(clf,X_train,y_train)
```

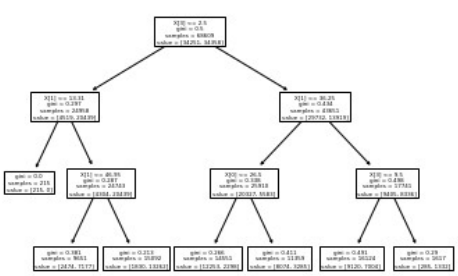
```
              precision    recall  f1-score   support

           0       0.71      0.86      0.78     14756
           1       0.82      0.64      0.72     14649

    accuracy                           0.75     29405
   macro avg       0.77      0.75      0.75     29405
weighted avg       0.77      0.75      0.75     29405

9371 1993
5278 12763
ROC  AUC value =  0.752

DecisionTreeClassifier(max_depth=3, random_state=1)
```

```python
prob = clf.predict_proba(X_test)[:,-1]
thresholds= [0.5,0.4,0.3,0.2,0.1]
tprs = []
fprs = []
for th in thresholds:
 y_pred = np.where(prob >= th, 1 , 0)
 tn, fp, fn, tp = confusion_matrix(y_test,y_pred).ravel()
 tpr = tp/(tp + fn)
 fpr = fp/(fp + tn)

 tprs.append(tpr)
 fprs.append(fpr)
plt.figure()
plt.plot(fprs,tprs,"x--")
plt.scatter(fprs,tprs)
plt.show()
```




```python
tree.plot_tree(clf)
```

```
[Text(141.64615384615385, 190.26, 'X[3] <= 2.5\ngini = 0.5\nsamples = 68609\nvalue = [34251, 34358]'),
 Text(51.50769230769231, 135.9, 'X[1] <= 13.31\ngini = 0.297\nsamples = 24958\nvalue = [4519, 20439]'),
 Text(25.753846153846155, 81.53999999999999, 'gini = 0.0\nsamples = 215\nvalue = [215, 0]'),
 Text(77.26153846153846, 81.53999999999999, 'X[1] <= 46.95\ngini = 0.287\nsamples = 24743\nvalue = [4304, 20439]'),
 Text(51.50769230769231, 27.180000000000007, 'gini = 0.381\nsamples = 9651\nvalue = [2474, 7177]'),
 Text(103.01538461538462, 27.180000000000007, 'gini = 0.213\nsamples = 15092\nvalue = [1830, 13262]'),
 Text(231.7846153846154, 135.9, 'X[1] <= 36.25\ngini = 0.434\nsamples = 43651\nvalue = [29732, 13919]'),
 Text(180.27692307692308, 81.53999999999999, 'X[0] <= 26.5\ngini = 0.338\nsamples = 25910\nvalue = [20327, 5583]'),
 Text(154.52307692307693, 27.180000000000007, 'gini = 0.266\nsamples = 14551\nvalue = [12253, 2298]'),
 Text(206.03076923076924, 27.180000000000007, 'gini = 0.411\nsamples = 11359\nvalue = [8074, 3285]'),
 Text(283.2923076923077, 81.53999999999999, 'X[3] <= 9.5\ngini = 0.498\nsamples = 17741\nvalue = [9405, 8336]'),
 Text(257.53846153846155, 27.180000000000007, 'gini = 0.491\nsamples = 16124\nvalue = [9120, 7004]'),
 Text(309.04615384615386, 27.180000000000007, 'gini = 0.29\nsamples = 1617\nvalue = [285, 1332]')]
```



```
# Model 11 RandomForest
rf=RandomForestClassifier(n_estimators=100)
create_model(rf,X_train,y_train)
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99     14756
           1       0.99      1.00      0.99     14649

    accuracy                           0.99     29405
   macro avg       0.99      0.99      0.99     29405
weighted avg       0.99      0.99      0.99     29405

14649 165
0 14591
ROC  AUC value =  0.994

RandomForestClassifier()
```

```
# since RandomForest has highest value of 0.994 it is selected for cross validation
from sklearn.model_selection import cross_val_score
rfm_scores = cross_val_score(rf,X_train,y_train,cv=20)
rfm_scores.mean()
```

```
0.9943884476387048
```

```
# since cross validation score is 0.994 is same as model value the model with Random Forrest algorithm
# is chosen as final solution
```

```
rf.feature_importances_
```

```
array([0.255805  , 0.22743417, 0.16197426, 0.12555099, 0.048403  ,
       0.00250893, 0.05151398, 0.12680966])
```

```
rf.feature_names = X_train.columns
rf.feature_names
```

```
Index(['Duration', 'Net Sales', 'Age', 'Agency', 'Agency Type',
       'Distribution Channel', 'Product Name', 'Destination'],
      dtype='object')
```

```
# Important features are 'Duration', 'Net Sales', 'Age', 'Agency' and 'Destination',
```