travel-insurance

May 26, 2024

About Dataset Context A Tour & Travels Company Is Offering Travel Insurance Package To Their Customers. The New Insurance Package Also Includes Covid Cover. The Company Requires To Know The Which Customers Would Be Interested To Buy It Based On Its Database History. The Insurance Was Offered To Some Of The Customers In 2019 And The Given Data Has Been Extracted From The Performance/Sales Of The Package During That Period. The Data Is Provided For Almost 2000 Of Its Previous Customers And You Are Required To Build An Intelligent Model That Can Predict If The Customer Will Be Interested To Buy The Travel Insurance Package Based On Certain Parameters Given Below. Image Credits-Unsplash(free to use)

Content 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 In Indian Rupees[Rounded To Nearest 50 Thousand Rupees] FamilyMembers- Number Of Members In Customer's Family ChronicDisease- Whether The Customer Suffers From Any Major Disease Or Conditions Like Diabetes/High BP or Asthama,etc. FrequentFlyer- Derived Data Based On Customer's History Of Booking Air Tickets On Atleast 4 Different Instances In The Last 2 Years[2017-2019]. EverTravelledAbroad- Has The Customer Ever Travelled To A Foreign Country[Not Necessarily Using The Company's Services] TravelInsurance- Did The Customer Buy Travel Insurance Package During Introductory Offering Held In The Year 2019.

[ ]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[ ]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from functions import *
from IPython.display import Image
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, roc_auc_score
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import GridSearchCV
```

[ ]:
```
path = "../travel-insurance-prediction/travel-insurance-dataset.csv"

travel_df = pd.read_csv(path)

travel_df.head()
```

[ ]:
```
   Unnamed: 0  Age               Employment Type GraduateOrNot  AnnualIncome  \
0           0   31             Government Sector           Yes        400000
1           1   31  Private Sector/Self Employed           Yes       1250000
2           2   34  Private Sector/Self Employed           Yes        500000
3           3   28  Private Sector/Self Employed           Yes        700000
4           4   28  Private Sector/Self Employed           Yes        700000

   FamilyMembers  ChronicDiseases FrequentFlyer EverTravelledAbroad  \
0              6                1            No                  No
1              7                0            No                  No
2              4                1            No                  No
3              3                1            No                  No
4              8                1           Yes                  No

   TravelInsurance
0                0
1                0
2                1
3                0
4                0
```

We can see that we have "Unnamed: 0" as a column name, which provides no value. Let's drop it.

[ ]:
```
travel_df = travel_df.drop("Unnamed: 0", axis=1, errors="ignore")
travel_df.head()
```

[ ]:
```
   Age               Employment Type GraduateOrNot  AnnualIncome  \
0   31             Government Sector           Yes        400000
1   31  Private Sector/Self Employed           Yes       1250000
2   34  Private Sector/Self Employed           Yes        500000
3   28  Private Sector/Self Employed           Yes        700000
4   28  Private Sector/Self Employed           Yes        700000

   FamilyMembers  ChronicDiseases FrequentFlyer EverTravelledAbroad  \
0              6                1            No                  No
1              7                0            No                  No
2              4                1            No                  No
```

```
3              3                 1           No                   No
4              8                 1          Yes                   No

   TravelInsurance
0                0
1                0
2                1
3                0
4                0
```

Now we can move into checking statistical summary of the numerical features in dataset.

[ ]:
```
travel_df.describe()
```

[ ]:
```
               Age   AnnualIncome  FamilyMembers  ChronicDiseases  \
count  1987.000000  1.987000e+03    1987.000000      1987.000000
mean     29.650226  9.327630e+05       4.752894         0.277806
std       2.913308  3.768557e+05       1.609650         0.448030
min      25.000000  3.000000e+05       2.000000         0.000000
25%      28.000000  6.000000e+05       4.000000         0.000000
50%      29.000000  9.000000e+05       5.000000         0.000000
75%      32.000000  1.250000e+06       6.000000         1.000000
max      35.000000  1.800000e+06       9.000000         1.000000

       TravelInsurance
count      1987.000000
mean          0.357323
std           0.479332
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max           1.000000
```

0.0.1 Dataset Numeric Features Overview

- **Total Records:** 1987 individuals
- **Variables:** Age, Annual Income (in Indian Rupees), Family Members, Chronic Diseases, Travel Insurance
- **Missing Values:** None in the numeric features.

0.0.2 Descriptive Statistics

- **Age:** Mean = 29.65, Range = 25 to 35 years
- **Annual Income (in Indian Rupees):** Mean = 932,763, Range = 300,000 to 1,800,000
- **Family Members:** Mean = 4.75, Range = 2 to 9
- **Chronic Diseases:** 27.78% have chronic diseases
- **Travel Insurance:** 35.73% have travel insurance

0.0.3 Key Observations

- **Income Variability:** Income varies widely, indicating potential impacts on travel insurance decisions.
- **Demographic Focus:** A focused age range (25-35) suggests a specific demographic possibly related to lifestyle choices.
- **Health Influence:** A significant minority have chronic diseases, influencing insurance needs.

Since I am making an analysis in Europe, I will convert the income(AnnualIncome feature) to Euro.

As of the time being 1 rupee is equivalent of 0.011 Euro

[ ]:
```
rupee_to_euro = 0.011

travel_df["AnnualIncomeEuro"] = travel_df["AnnualIncome"] * rupee_to_euro
travel_df.head()
```

[ ]:
```
   Age               Employment Type GraduateOrNot  AnnualIncome  \
0   31             Government Sector           Yes        400000
1   31  Private Sector/Self Employed           Yes       1250000
2   34  Private Sector/Self Employed           Yes        500000
3   28  Private Sector/Self Employed           Yes        700000
4   28  Private Sector/Self Employed           Yes        700000

   FamilyMembers  ChronicDiseases FrequentFlyer EverTravelledAbroad  \
0              6                1            No                  No
1              7                0            No                  No
2              4                1            No                  No
3              3                1            No                  No
4              8                1           Yes                  No

   TravelInsurance  AnnualIncomeEuro
0                0            4400.0
1                0           13750.0
2                1            5500.0
3                0            7700.0
4                0            7700.0
```

We added a new feature called **AnnualIncomeEuro** to show incomes in Euros. This makes the data clearer.

We might remove the old **AnnualIncome** feature to keep everything consistent.

First, we need to check the data types and make sure there are no missing values in the new feature.

[ ]:
```
travel_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1987 entries, 0 to 1986
Data columns (total 10 columns):
```

```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  1987 non-null   int64
 1   Employment Type      1987 non-null   object
 2   GraduateOrNot        1987 non-null   object
 3   AnnualIncome         1987 non-null   int64
 4   FamilyMembers        1987 non-null   int64
 5   ChronicDiseases      1987 non-null   int64
 6   FrequentFlyer        1987 non-null   object
 7   EverTravelledAbroad  1987 non-null   object
 8   TravelInsurance      1987 non-null   int64
 9   AnnualIncomeEuro     1987 non-null   float64
dtypes: float64(1), int64(5), object(4)
memory usage: 155.4+ KB
```

We can see that there are no missing values in the dataset, therefore we can proceed with dropping the **AnnualIncome** feature, which represents the income in rupees.

[ ]:
```
travel_df = travel_df.drop("AnnualIncome", axis=1, errors="ignore")
travel_df.head()
```

[ ]:
```
   Age               Employment Type GraduateOrNot  FamilyMembers  \
0   31             Government Sector           Yes              6
1   31  Private Sector/Self Employed           Yes              7
2   34  Private Sector/Self Employed           Yes              4
3   28  Private Sector/Self Employed           Yes              3
4   28  Private Sector/Self Employed           Yes              8

   ChronicDiseases FrequentFlyer EverTravelledAbroad  TravelInsurance  \
0                1            No                  No                0
1                0            No                  No                0
2                1            No                  No                1
3                1            No                  No                0
4                1           Yes                  No                0

   AnnualIncomeEuro
0            4400.0
1           13750.0
2            5500.0
3            7700.0
4            7700.0
```

Next, lets quickly fix the naming of the **Employment Type** feature. from **Employment Type** to **EmploymentType**.

[ ]:
```
travel_df = travel_df.rename(columns={"Employment Type": "EmploymentType"})
travel_df.head()
```

```
[ ]:    Age                EmploymentType GraduateOrNot  FamilyMembers  \
     0   31            Government Sector           Yes              6
     1   31  Private Sector/Self Employed           Yes              7
     2   34  Private Sector/Self Employed           Yes              4
     3   28  Private Sector/Self Employed           Yes              3
     4   28  Private Sector/Self Employed           Yes              8

        ChronicDiseases FrequentFlyer EverTravelledAbroad  TravelInsurance  \
     0                1            No                  No                0
     1                0            No                  No                0
     2                1            No                  No                1
     3                1            No                  No                0
     4                1           Yes                  No                0

        AnnualIncomeEuro
     0            4400.0
     1           13750.0
     2            5500.0
     3            7700.0
     4            7700.0
```

As we have fixed the column names, we can move on to fixing some of the features for consistency in our visuals further, when we will be checking for distributions, such as **EmploymentType**, **FrequentFlyer**, **EverTravelledAbroad**.

So modifications will be as follows:

- 1 for **Government Sector** 0 for **Private Sector/Self Employed** in **EmploymentType**
- 1 for **Yes** 0 for **No** in **FrequentFlyer** and **EverTravelledAbroad**

```
[ ]: travel_df["GraduateOrNot"] = travel_df["GraduateOrNot"].map({"Yes": 1, "No": 0})
travel_df["FrequentFlyer"] = travel_df["FrequentFlyer"].map({"Yes": 1, "No": 0})
travel_df["EverTravelledAbroad"] = travel_df["EverTravelledAbroad"].map(
    {"Yes": 1, "No": 0}
)

travel_df["EmploymentType"] = travel_df["EmploymentType"].map(
    {"Government Sector": 1, "Private Sector/Self Employed": 0}
)
travel_df.head()
```

```
[ ]:    Age  EmploymentType  GraduateOrNot  FamilyMembers  ChronicDiseases  \
     0   31               1              1              6                1
     1   31               0              1              7                0
     2   34               0              1              4                1
     3   28               0              1              3                1
     4   28               0              1              8                1

        FrequentFlyer  EverTravelledAbroad  TravelInsurance  AnnualIncomeEuro
```

0	0	0	0	4400.0
1	0	0	0	13750.0
2	0	0	1	5500.0
3	0	0	0	7700.0
4	1	0	0	7700.0

[ ]:
```
numerical_features = ["Age", "FamilyMembers", "AnnualIncomeEuro"]
categorical_features_one = [
    "EmploymentType",
    "GraduateOrNot",
    "FrequentFlyer",
]
categorical_features_two = [
    "EverTravelledAbroad",
    "TravelInsurance",
    "ChronicDiseases",
]

categorical_features = categorical_features_one + categorical_features_two
```

[ ]:
```
plot_combined_histograms(
    travel_df, numerical_features, save_path="images/combined_histograms.png"
)
```

[ ]:
```
%%capture
Image(filename="images/combined_histograms.png")
```

0.0.4 Age

- **Distribution**: The age distribution shows a significant concentration around 30 years.
- **Range**: Ages in the dataset range from 25 to 35 years.
- **Observations**: There is a noticeable peak at age 30, indicating a higher number of individuals in this age group.

0.0.5 Family Members

- **Distribution**: The distribution of family members shows a central peak.
- **Range**: Family sizes range from 2 to 9 members.
- **Observations**: Most individuals have 4 to 6 family members, with 4 being the most common family size.

0.0.6 Annual Income (Euro)

- **Distribution**: The annual income distribution is quite spread out with multiple peaks.
- **Range**: Annual incomes range from approximately 4,400 to 20,000 Euros.
- **Observations**: The data shows several peaks, particularly around 5,000, 10,000, and 15,000 Euros, indicating diverse income levels among the individuals.

Now we can move to bar charts for our categorical features.

```
[ ]: plot_combined_bar_charts(
         travel_df, categorical_features_one, save_path="images/combined_bar_charts1.
      ↪png"
     )
```

```
[ ]: %%capture
     Image(filename="images/combined_bar_charts1_part1.png")
```

Employment Type: The higher proportion of individuals in the **Private Sector/Self Employed** category might influence their travel behavior and insurance uptake differently compared to those in the Government Sector. This could be due to factors such as job stability and disposable income.

Education Level: The overwhelming majority of graduates in the dataset could suggest that educational attainment might be a significant factor in travel insurance decisions. Graduates might have different travel patterns and risk perceptions compared to non-graduates.

Travel Frequency: The fact that most individuals are not frequent flyers might impact the likelihood of purchasing travel insurance. Frequent flyers could have a higher propensity to buy travel insurance due to increased travel exposure and associated risks

```
[ ]: plot_combined_bar_charts(
         travel_df, categorical_features_two, save_path="images/combined_bar_charts2.
      ↪png"
     )
```

```
[ ]: %%capture
     Image(filename="images/combined_bar_charts2_part1.png")
```

International Travel Experience: The significantly higher number of individuals who have not traveled abroad might affect their perception of the need for travel insurance. Those with international travel experience may have different risk assessments and insurance requirements.

Travel Insurance Adoption: The lower proportion of individuals with travel insurance highlights an opportunity for increasing travel insurance coverage. Understanding the factors that influence the decision to purchase travel insurance will be crucial for targeted marketing and product development.

Health Conditions: The distribution of chronic diseases can provide insights into health-related considerations that might influence travel insurance decisions. Individuals with chronic diseases might have different requirements and motivations for purchasing insurance.

Now lets move on to the outliers, lets see if our numerical features has any outliers.

```
[ ]: plot_combined_boxplots(
         travel_df, numerical_features, save_path="images/combined_boxplot.png"
     )
```

```
[ ]: %%capture
     Image(filename="images/combined_boxplot.png")
```

1. **Age**:

    - **Range**: The ages of individuals range from 26 to 34 years.
    - **Median**: The median age is 30 years.
    - **Distribution**: Age is fairly evenly spread around the median, with no significant outliers.

2. **Family Members**:

    - **Range**: The number of family members ranges from 2 to 9.
    - **Median**: The median number of family members is 5.
    - **Distribution**: The distribution is slightly skewed to the right, indicating a few families with more members but generally centered around 4 to 6 members. No significant outliers are present.

3. **Annual Income (in Euros)**:

    - **Range**: Annual income ranges from 4,000 to 20,000 Euros.
    - **Median**: The median annual income is 10,000 Euros.
    - **Distribution**: The income distribution is fairly symmetrical, centered around the median, with no significant outliers.

Lets confirm that there are no anomalies via the IQR method.

[ ]:
```
detect_anomalies_iqr(travel_df, numerical_features)
```

```
No anomalies detected in feature 'Age'.
No anomalies detected in feature 'FamilyMembers'.
No anomalies detected in feature 'AnnualIncomeEuro'.
```

[ ]:
```
Empty DataFrame
Columns: [Age, FamilyMembers, AnnualIncomeEuro]
Index: []
```

We can confirm that there are no anomalies in our numerical features. Now we can move on to the correlation matrix, to check for multicollinearity.

[ ]:
```
plot_correlation_matrix(
    travel_df, numerical_features, save_path="images/correlation_matrix.png"
)
```

[ ]:
```
%%capture
Image(filename="images/correlation_matrix.png")
```

Here is the correlation analysis of the numerical features in the dataset:

Age:

- **Correlation with FamilyMembers: 0.027** - This indicates a very weak positive correlation between age and the number of family members.
- **Correlation with AnnualIncomeEuro: -0.020** - This indicates a very weak negative correlation between age and annual income in Euros.

FamilyMembers:

- **Correlation with Age: 0.027** - As mentioned above, there is a very weak positive correlation.
- **Correlation with AnnualIncomeEuro: -0.015** - This indicates a very weak negative correlation between the number of family members and annual income in Euros.

AnnualIncomeEuro:

- **Correlation with Age: -0.020** - As mentioned above, there is a very weak negative correlation.
- **Correlation with FamilyMembers: -0.015** - As mentioned above, there is a very weak negative correlation.

These correlations are all very weak, indicating that there is little linear relationship between these numerical features.

Next, we examine how each categorical feature correlates with the target variable **TravelInsurance**. This helps us understand which features might be most relevant in predicting whether an individual has travel insurance. For this, we perform Chi-Square tests.

The Chi-Square test is used to determine if there is a significant association between two categorical variables. In our case, we use it to check the relationship between various categorical features and the target variable (`TravelInsurance`).

- **Null Hypothesis (H0)**: There is no association between the categorical feature and the target variable.
- **Alternative Hypothesis (H1)**: There is an association between the categorical feature and the target variable.

Our target population for this analysis consists of customers who were offered a travel insurance package by a tour and travels company in 2019.

[ ]:
```
target_feature = "TravelInsurance"
categorical_features_test = [
    "EmploymentType",
    "GraduateOrNot",
    "FrequentFlyer",
    "EverTravelledAbroad",
    "ChronicDiseases",
]

chi_square_test(travel_df, categorical_features_test, target_feature)
```

```
Chi-Square test results for 'EmploymentType':
Chi2 statistic = 42.75380328896317, p-value = 6.208106601512192e-11
Significant association between 'EmploymentType' and 'TravelInsurance'.

Chi-Square test results for 'GraduateOrNot':
Chi2 statistic = 0.605510042905726, p-value = 0.4364833842842336
No significant association between 'GraduateOrNot' and 'TravelInsurance'.

Chi-Square test results for 'FrequentFlyer':
```

```
Chi2 statistic = 105.85723074203977, p-value = 7.924360415064537e-25
Significant association between 'FrequentFlyer' and 'TravelInsurance'.

Chi-Square test results for 'EverTravelledAbroad':
Chi2 statistic = 370.5599281861554, p-value = 1.4134505859999571e-82
Significant association between 'EverTravelledAbroad' and 'TravelInsurance'.

Chi-Square test results for 'ChronicDiseases':
Chi2 statistic = 0.5754114650274649, p-value = 0.4481165216392011
No significant association between 'ChronicDiseases' and 'TravelInsurance'.
```

The Chi-Square test was conducted to determine the association between several categorical features and the target variable `TravelInsurance`. Here are the results, including the hypotheses and conclusions for each feature:

0.0.7 EmploymentType

- **Null Hypothesis (H0)**: There is no association between `EmploymentType` and `TravelInsurance`.
- **Alternative Hypothesis (H1)**: There is an association between `EmploymentType` and `TravelInsurance`.
- **Chi2 Statistic**: 42.75
- **P-Value**: 6.21e-11
- **Conclusion**: Since the p-value is less than 0.05, we reject the null hypothesis. There is a significant association between `EmploymentType` and `TravelInsurance`. This indicates that the sector in which a customer is employed influences the likelihood of purchasing travel insurance.

0.0.8 GraduateOrNot

- **Null Hypothesis (H0)**: There is no association between `GraduateOrNot` and `TravelInsurance`.
- **Alternative Hypothesis (H1)**: There is an association between `GraduateOrNot` and `TravelInsurance`.
- **Chi2 Statistic**: 0.61
- **P-Value**: 0.44
- **Conclusion**: Since the p-value is greater than 0.05, we fail to reject the null hypothesis. There is no significant association between `GraduateOrNot` and `TravelInsurance`. This suggests that whether a customer is a college graduate does not significantly influence their decision to purchase travel insurance.

0.0.9 FrequentFlyer

- **Null Hypothesis (H0)**: There is no association between `FrequentFlyer` and `TravelInsurance`.
- **Alternative Hypothesis (H1)**: There is an association between `FrequentFlyer` and `TravelInsurance`.
- **Chi2 Statistic**: 105.86
- **P-Value**: 7.92e-25

- **Conclusion**: Since the p-value is less than 0.05, we reject the null hypothesis. There is a significant association between `FrequentFlyer` and `TravelInsurance`. This indicates that customers who are frequent flyers are more likely to purchase travel insurance.

0.0.10 EverTravelledAbroad

- **Null Hypothesis (H0)**: There is no association between `EverTravelledAbroad` and `TravelInsurance`.
- **Alternative Hypothesis (H1)**: There is an association between `EverTravelledAbroad` and `TravelInsurance`.
- **Chi2 Statistic**: 370.56
- **P-Value**: 1.41e-82
- **Conclusion**: Since the p-value is less than 0.05, we reject the null hypothesis. There is a significant association between `EverTravelledAbroad` and `TravelInsurance`. Customers who have traveled abroad are significantly more likely to buy travel insurance.

0.0.11 ChronicDiseases

- **Null Hypothesis (H0)**: There is no association between `ChronicDiseases` and `TravelInsurance`.
- **Alternative Hypothesis (H1)**: There is an association between `ChronicDiseases` and `TravelInsurance`.
- **Chi2 Statistic**: 0.58
- **P-Value**: 0.45
- **Conclusion**: Since the p-value is greater than 0.05, we fail to reject the null hypothesis. There is no significant association between `ChronicDiseases` and `TravelInsurance`. Having a chronic disease does not significantly impact the decision to purchase travel insurance.

Out target population remains the same, which consists of customers who were offered a travel insurance package by a tour and travels company in 2019.

We move on to checking the following hypothesis tests:

Hypothesis 1: Age

- **Null Hypothesis (H0)**: There is no difference in the mean age between customers who purchased travel insurance and those who did not.
- **Alternative Hypothesis (H1)**: There is a difference in the mean age between customers who purchased travel insurance and those who did not.

Hypothesis 2: Annual Income (Euro)

- **Null Hypothesis (H0)**: There is no difference in the mean annual income (in Euros) between customers who purchased travel insurance and those who did not.
- **Alternative Hypothesis (H1)**: There is a difference in the mean annual income (in Euros) between customers who purchased travel insurance and those who did not.

Hypothesis 3: Family Members

- **Null Hypothesis (H0)**: There is no difference in the mean number of family members between customers who purchased travel insurance and those who did not.

- **Alternative Hypothesis (H1)**: There is a difference in the mean number of family members between customers who purchased travel insurance and those who did not.

So, lets calculate confidence intervals for important metrics (e.g., average age, average income). This provides an estimate of the population parameter with a specified level of confidence (95% in our case).

[ ]:
```
analyze_features(travel_df, numerical_features, target_feature)
```

```
95% confidence interval for Age (insured): (29.643243997358706,
30.13422079137369)
95% confidence interval for Age (not insured): (29.372444588596146,
29.66279425243753)
95% confidence interval for FamilyMembers (insured): (4.8016914431315465,
5.049012782220566)
95% confidence interval for FamilyMembers (not insured): (4.5712214858818285,
4.7427957419960105)
95% confidence interval for AnnualIncomeEuro (insured): (12161.821774091104,
12769.445831542698)
95% confidence interval for AnnualIncomeEuro (not insured): (8835.680551388663,
9232.917725823552)
95% confidence interval for AnnualIncomeEuro (insured): (12161.821774091104,
12769.445831542698)
95% confidence interval for AnnualIncomeEuro (not insured): (8835.680551388663,
9232.917725823552)
```

So our results show the following:

Age

- **95% confidence interval for Age (insured):** (29.64, 30.13)
- **95% confidence interval for Age (not insured):** (29.37, 29.66)
- **Interpretation:** The mean age of customers who purchased travel insurance is estimated to be between 29.64 and 30.13 years with 95% confidence. For those who did not purchase travel insurance, the mean age is estimated to be between 29.37 and 29.66 years with 95% confidence. Since the confidence intervals overlap, this suggests that there might not be a significant difference in the mean ages of the two groups.

Family Members

- **95% confidence interval for FamilyMembers (insured):** (4.80, 5.05)
- **95% confidence interval for FamilyMembers (not insured):** (4.57, 4.74)
- **Interpretation:** The mean number of family members for customers who purchased travel insurance is estimated to be between 4.80 and 5.05 with 95% confidence. For those who did not purchase travel insurance, the mean number of family members is estimated to be between 4.57 and 4.74 with 95% confidence. The lack of overlap in the confidence intervals suggests a potential significant difference in the number of family members between the two groups.

Annual Income (Euro)

- **95% confidence interval for AnnualIncomeEuro (insured):** (12161.82, 12769.45)

- **95% confidence interval for AnnualIncomeEuro (not insured):** (8835.68, 9232.92)
- **Interpretation:** The mean annual income in Euros for customers who purchased travel insurance is estimated to be between 12161.82 and 12769.45 with 95% confidence. For those who did not purchase travel insurance, the mean annual income is estimated to be between 8835.68 and 9232.92 with 95% confidence. The lack of overlap in the confidence intervals indicates a significant difference in the annual income between the two groups.

With these results of our confidence interval, lets move to our hypothesis testing, via **Mann-Whitney U test**, as our numerical features are not normally distributed.

[ ]:
```
analyze_mannwhitneyu(travel_df, numerical_features, target_feature)
```

```
Mann-Whitney U test for Age: U-statistic = 479515.0, p-value =
0.030678780881604327
Significant difference in distributions for Age.
Mann-Whitney U test for FamilyMembers: U-statistic = 493531.5, p-value =
0.0008301993194480642
Significant difference in distributions for FamilyMembers.
Mann-Whitney U test for AnnualIncomeEuro: U-statistic = 670230.5, p-value =
3.020101045488567e-70
Significant difference in distributions for AnnualIncomeEuro.
```

Age

- **U-Statistic**: 479515.0
- **P-Value**: 0.0307
- **Conclusion**: Since the p-value is less than 0.05, we reject the null hypothesis (H0). There is a significant difference in the mean age between customers who purchased travel insurance and those who did not. This supports Hypothesis 1 (H1), indicating that age plays a role in the decision to purchase travel insurance.

Family Members

- **U-Statistic**: 493531.5
- **P-Value**: 0.0008
- **Conclusion**: Since the p-value is less than 0.05, we reject the null hypothesis (H0). There is a significant difference in the mean number of family members between customers who purchased travel insurance and those who did not. This supports Hypothesis 3 (H1), suggesting that the number of family members influences the likelihood of purchasing travel insurance.

Annual Income (Euro)

- **U-Statistic**: 670230.5
- **P-Value**: 3.02e-70
- **Conclusion**: Since the p-value is less than 0.05, we reject the null hypothesis (H0). There is a significant difference in the mean annual income (in Euros) between customers who purchased travel insurance and those who did not. This supports Hypothesis 2 (H1), indicating that annual income is a significant factor in the decision to buy travel insurance.

Now we can move on to our machine learning models. We will start by scaling the numerical features to normalize the data distribution. There is no need to scale the categorical features, as they have already been encoded.

[ ]:
```
scaler = StandardScaler()
travel_df[numerical_features] = scaler.
    ↪fit_transform(travel_df[numerical_features])
```

Now we can split the data into training and testing sets.

[ ]:
```
X = travel_df.drop('TravelInsurance', axis=1)
y = travel_df['TravelInsurance']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
    ↪random_state=42)
```

We will no select a variety of machine learning models to apply. This will include **logistic regression, random forest, gradient boosting, and support vector machine**

[ ]:
```
models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(),
    'SVM': SVC(probability=True)
}
```

Now we will train the models on the training data and evaluate their performance on the testing data.

[ ]:
```
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:, 1]

    results[name] = {
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1 Score': f1_score(y_test, y_pred),
        'ROC AUC': roc_auc_score(y_test, y_proba)
    }

for model_name, metrics in results.items():
    print(f"\n{model_name} Performance:")
    for metric, value in metrics.items():
        print(f"{metric}: {value:.4f}")
```

```
Logistic Regression Performance:
Accuracy: 0.7688
```

```
Precision: 0.7692
Recall: 0.4965
F1 Score: 0.6034
ROC AUC: 0.7369

Random Forest Performance:
Accuracy: 0.8116
Precision: 0.7895
Recall: 0.6383
F1 Score: 0.7059
ROC AUC: 0.7833

Gradient Boosting Performance:
Accuracy: 0.8442
Precision: 0.9877
Recall: 0.5674
F1 Score: 0.7207
ROC AUC: 0.8064

SVM Performance:
Accuracy: 0.8116
Precision: 0.8667
Recall: 0.5532
F1 Score: 0.6753
ROC AUC: 0.8000
```

Based on the results above, we can see that **Gradient Boosting** achieved the highest accuracy (0.8442) and a good balance between precision (0.9877) and recall (0.5674), with an ROC AUC of 0.8064.

Random Forest also performed well with an accuracy of 0.8116 and balanced precision (0.7895) and recall (0.6383), with an ROC AUC of 0.7833.

Logistic Regression had moderate performance with lower recall, which indicates it might miss predicting some positive cases.

SVM had a good accuracy (0.8116) and high precision (0.8667) but lower recall (0.5532).

To further improve the performance, especially to balance precision and recall, we will be using an ensemble method like a voting classifier.

[ ]:
```
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30]
}

grid_search_rf = GridSearchCV(RandomForestClassifier(), param_grid_rf, cv=5,␣
 ↪scoring='roc_auc')
grid_search_rf.fit(X_train, y_train)
```

```
print("Best parameters found: ", grid_search_rf.best_params_)
print("Best ROC AUC score: ", grid_search_rf.best_score_)
```

```
Best parameters found:  {'max_depth': 10, 'n_estimators': 200}
Best ROC AUC score:  0.8107482016250065
```

[ ]:

```
voting_clf = VotingClassifier(
    estimators=[
        ('lr', LogisticRegression()),
        ('rf', RandomForestClassifier(n_estimators=grid_search.
 ↪best_params_['n_estimators'], max_depth=grid_search.
 ↪best_params_['max_depth'])),
        ('gb', GradientBoostingClassifier())
    ],
    voting='soft'
)

voting_clf.fit(X_train, y_train)
y_pred_ensemble = voting_clf.predict(X_test)
y_proba_ensemble = voting_clf.predict_proba(X_test)[:, 1]

ensemble_results = {
    'Accuracy': accuracy_score(y_test, y_pred_ensemble),
    'Precision': precision_score(y_test, y_pred_ensemble),
    'Recall': recall_score(y_test, y_pred_ensemble),
    'F1 Score': f1_score(y_test, y_pred_ensemble),
    'ROC AUC': roc_auc_score(y_test, y_proba_ensemble)
}

for metric, value in ensemble_results.items():
    print(f"{metric}: {value:.4f}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[74], line 4
      1 voting_clf = VotingClassifier(
      2     estimators=[
      3         ('lr', LogisticRegression()),
----> 4         ('rf', RandomForestClassifier(n_estimators=grid_search.
 ↪best_params_['n_estimators'], max_depth=grid_search.
 ↪best_params_['max_depth'])),
      5         ('gb', GradientBoostingClassifier())
      6     ],
      7     voting='soft'
      8 )
     10 voting_clf.fit(X_train, y_train)
     11 y_pred_ensemble = voting_clf.predict(X_test)
```

```
NameError: name 'grid_search' is not defined
```

[ ]:
```
%run -i functions.py
```

[ ]:
```

```