Final Project - Travel Insurance Predictions Team 7

September 27, 2021

ADS-505 Final Project - Travel Insurance Predictions

Team: #7

Team Members: Jimmy Nguyen, Christopher Robinson, Nima Amin Taghavi

Date: 09/20/2021

Programmin Language: Python Code

Table of Contents

---

1 Problem statement

About the Client

The client in this data mining project is a tour & travels company that is offering travel insurance package to their customers. This new insurance package also includes COVID-19 coverage for their flights. However, the client wants to know which customers based on their data base history are potential purchasers who may be interested in buying this new insurance package. Previously, the insurance package was offered to some of the customers in 2019 and data was collected from the performance and sales of the package during that period. The sample data given has close to 2000 customers from that period. The client is requesting information on which customer are most likely going to buy the travel insurance given their information such as employment type, income level, etc.

Business Problem

The client may use the solutions presented to them for customer-targeted advertising of the new travel insurance package. Also, data visualizations provided will help derive interesting insights about their potential buyers in order to optimize marketing strategies.

Data Mining Problem

- A supervised classification task, where the outcome variable of interest is *TravelInsurance* that indicates whether the customer will buy the travel insurance. Performance metrics should take in consideration the positive class of buyers/purchasers.
- Find out interesting patterns and trends for better customer segmentations through data exploration and visualizations.
- An unsupervised task, where the goal is to cluster customers.

---

2 Packages

Python code:

[1]:

```
%%javascript
IPython.OutputArea.prototype._should_scroll = function(lines) {
    return false;
}
```

```
<IPython.core.display.Javascript object>
```

[2]:

```
from pathlib import Path
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression, LogisticRegression,␣
 ↪LogisticRegressionCV
from sklearn.model_selection import train_test_split
```

```python
import statsmodels.api as sm
import scikitplot as skplt
from mord import LogisticIT
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score, GridSearchCV
from dmba import regressionSummary, stepwise_selection
from dmba import forward_selection, backward_elimination, exhaustive_search
from dmba import classificationSummary, gainsChart, liftChart
from dmba.metric import AIC_score
from tabulate import tabulate
import matplotlib.patches as mpatches
import warnings
sns.set_theme()
plt.rcParams['figure.figsize'] = [11, 9]

warnings.filterwarnings('ignore')
```

3 Data Set

Data Dictionary

1. **Age** - Age Of The Customer
2. **Employment Type** - The Sector In Which Customer Is Employed
3. **GraduateOrNot** - Whether The Customer Is College Graduate Or Not
4. **AnnualIncome** - The Yearly Income Of The Customer In Indian Rupees[Rounded To Nearest 50 Thousand Rupees]
5. **FamilyMembers** - Number Of Members In Customer's Family
6. **ChronicDisease** - Whether The Customer Suffers From Any Major Disease Or Conditions Like Diabetes/High BP or Asthama,etc.
7. **FrequentFlyer** - Derived Data Based On Customer's History Of Booking Air Tickets On Atleast 4 Different Instances In The Last 2 Years[2017-2019].
8. **EverTravelledAbroad** - Has The Customer Ever Travelled To A Foreign Country[Not Necessarily Using The Company's Services]
9. **TravelInsurance** - Did The Customer Buy Travel Insurance Package During Introductory Offering Held In The Year 2019.

Python code:

```
[3]: # Load data set
     df = pd.read_csv("../../Data/TravelInsurancePrediction.csv")

     # First few rows of data set
     df.head(3)
```

```
[3]:    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

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

        TravelInsurance
     0                0
     1                0
     2                1
```

---

4 Exploratory Data Analysis (EDA)

- Graphical and non-graphical representations of relationships between the response variable and predictor variables

4.1 Examing customers' Age

- Age distributions
- Age with Target Variable Overlaid
- Normalized Histogram with Target Overlaid on Age
- Age Group Comparisons (20s vs. 30s)
- Percentage of Purchases between Age groups (20s vs. 30s)

4.1.1 Age Distribution

Python code:

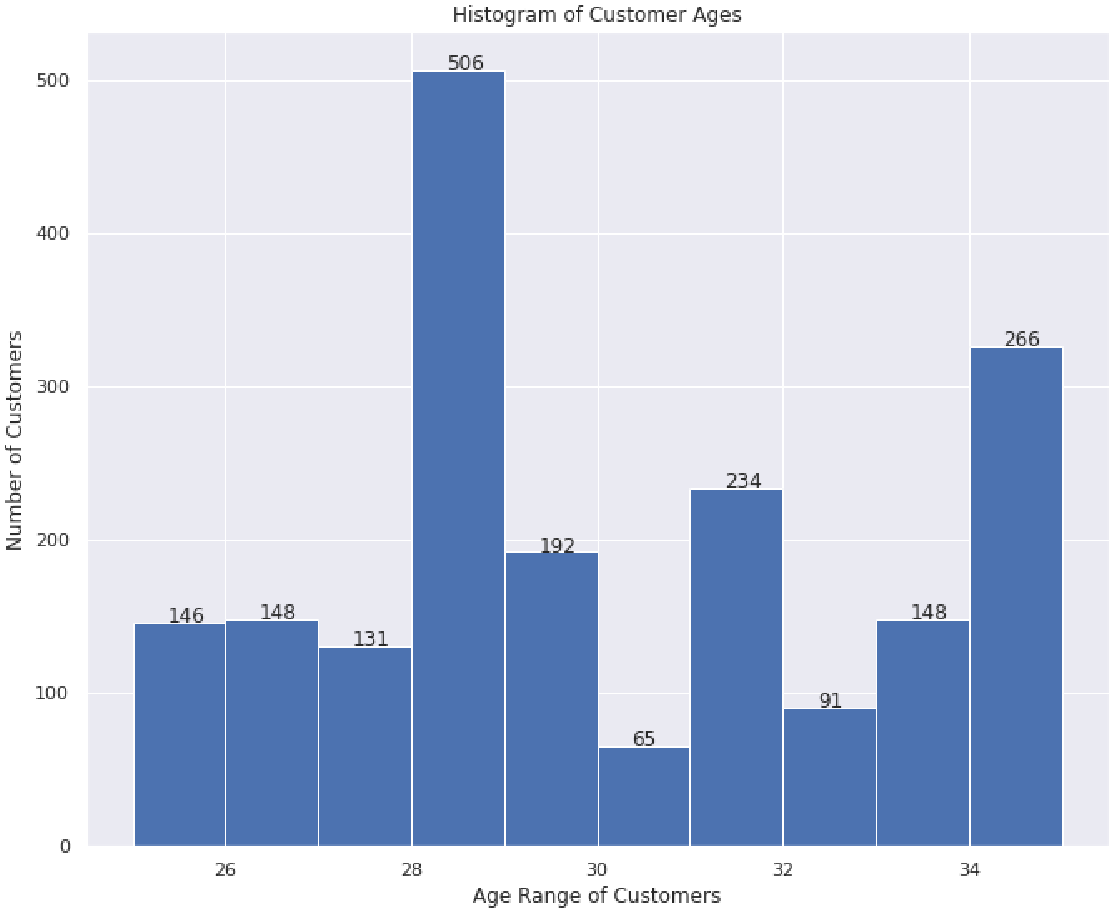
```
[4]: # Get a range of customer ages
     age = pd.DataFrame({'Age': df['Age'].value_counts().sort_index()})
     age
```

```
[4]:     Age
     25  146
     26  148
     27  131
```

```
28  506
29  192
30   65
31  234
32   91
33  148
34  266
35   60
```

[5]:

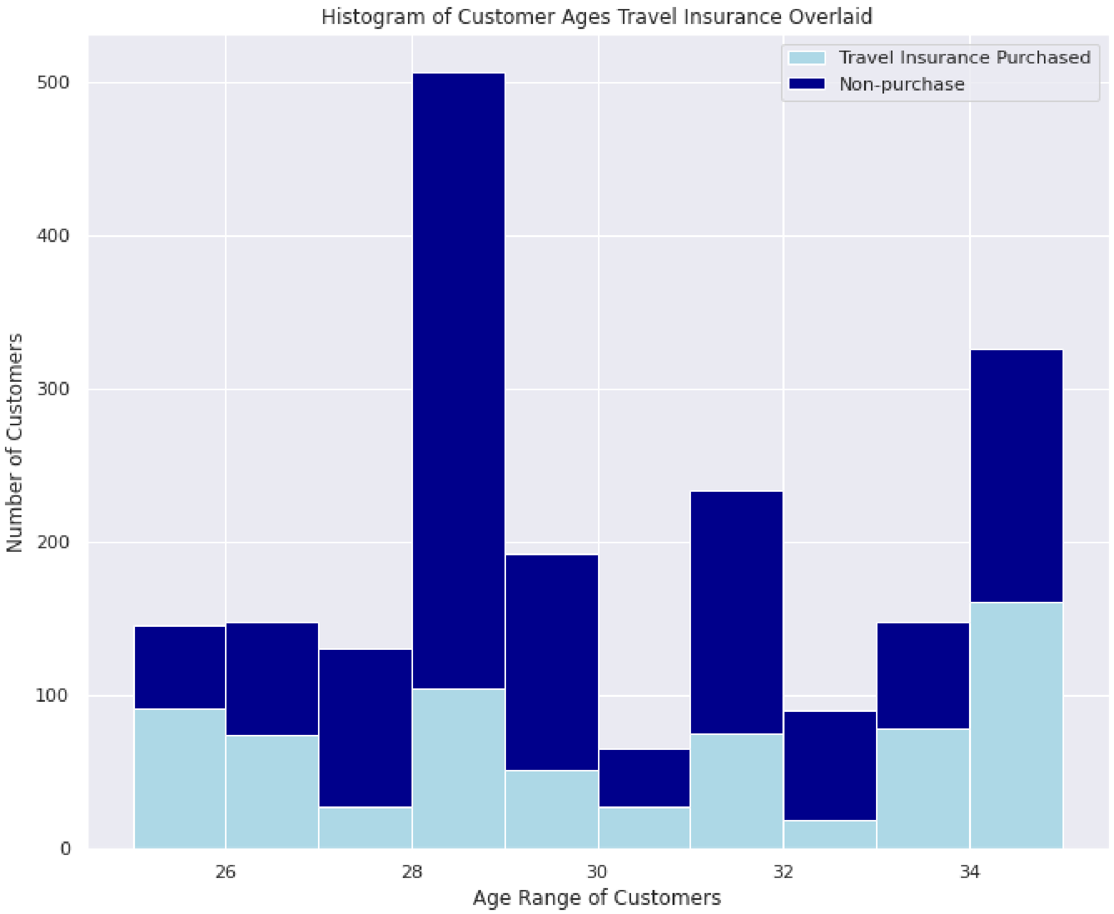
```
# Histogram of Age and set the range of bins from 25-35
bins = np.arange(25, 36)
ax = df['Age'].plot.hist(bins=bins)

# add labels
for p, label in zip(ax.patches, df['Age'].value_counts().sort_index()):
    ax.annotate(label, (p.get_x() + 0.37, p.get_height() + 0.15))

# title and axis
plt.title("Histogram of Customer Ages")
plt.xlabel("Age Range of Customers")
plt.ylabel("Number of Customers")
plt.show()
```



4.1.2 Age with Target Variable Overlaid

Python code:

```
[6]: # Set up plot with response overlaid
n, bins, patches = plt.hist(
    [
        df[df['TravelInsurance'] == 1]['Age'],
        df[df['TravelInsurance'] == 0]['Age']
    ],
    bins=10,
    stacked=True,
    color=["lightblue", "darkblue"],
)

# title and axis
labels = ["Travel Insurance Purchased", "Non-purchase"]
plt.legend(labels)
plt.title("Histogram of Customer Ages Travel Insurance Overlaid")
plt.xlabel("Age Range of Customers")
plt.ylabel('Number of Customers')
plt.show()
```

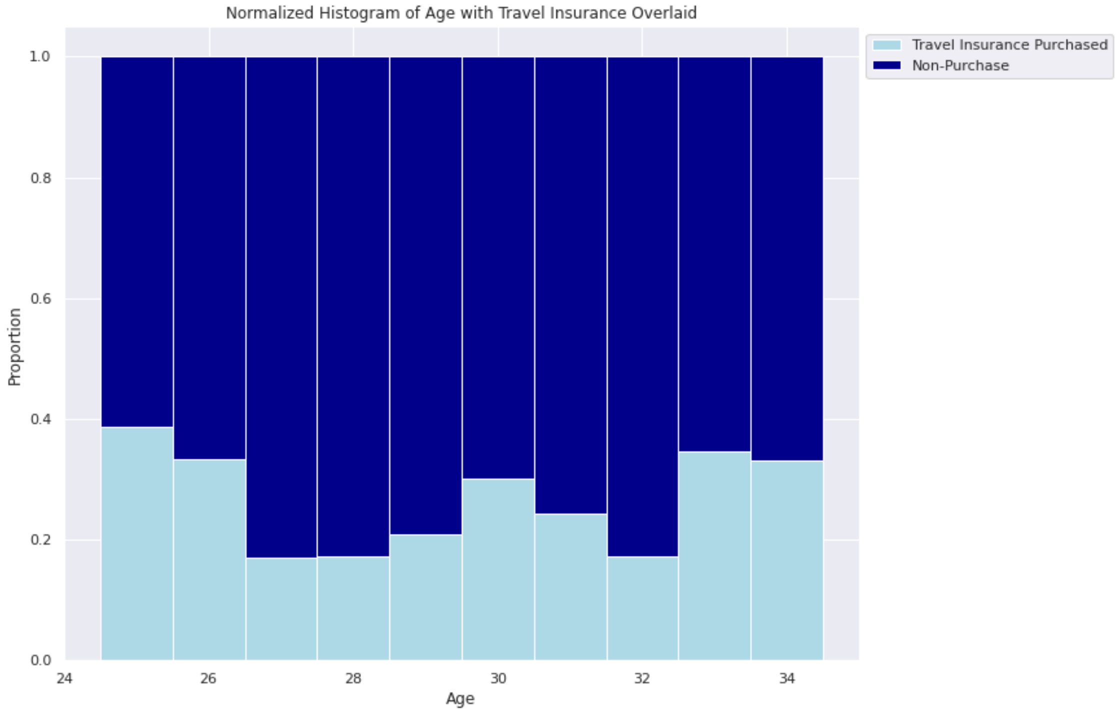


4.1.3 Normalized Histogram with Target Variable Overlaid on Age

Python code:

```
[7]: # Create normalized histogram for age groups by target overlay
n_table = np.column_stack((n[0], n[1]))  # stack the tables
n_norm = n_table / n_table.sum(axis=1)[:,
                                       None]  # create normalized tables by sum
ourbins = np.column_stack((bins[0:10], bins[1:11]))  # create table bins

p1 = plt.bar(x=ourbins[:, 0],
             height=n_norm[:, 0],
             width=ourbins[:, 1] - ourbins[:, 0],color = "lightblue")  # first␣
 ↪bar chart
```

```
p2 = plt.bar(
    x=ourbins[:, 0],
    height=n_norm[:, 1],
    width=ourbins[:, 1] - ourbins[:, 0],  # second bar chart
    bottom=n_norm[:, 0], color = "darkblue")
# Annotate legend, title with x and y labels
plt.legend(['Travel Insurance Purchased', 'Non-Purchase'],
           bbox_to_anchor=(1, 1))
plt.title('Normalized Histogram of Age with Travel Insurance Overlaid')
plt.xlabel('Age')
plt.ylabel('Proportion')
plt.show()
```

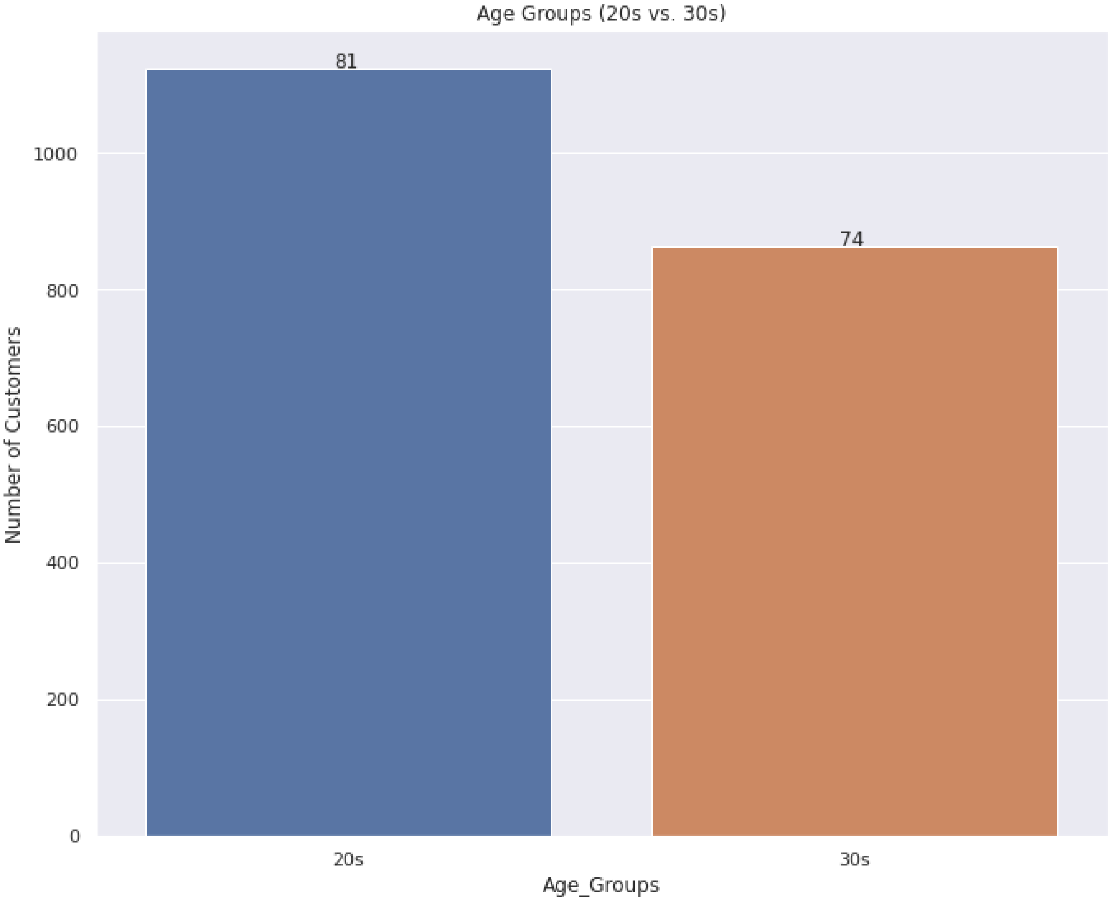


4.1.4 Age Groups Comparison (20s vs. 30s)

Python code:

[8]:

```
# Create function to categorize age groups
def age_groups(x):
    '''
    x: This is a value from df['Age']
    returns each as a new categorical value of 20s or 30s
    '''
    if x < 30:
        return '20s'
    else:
        return '30s'


# Apply age_groups function on each value
age_groups = pd.DataFrame(
    {'Age_Groups': df['Age'].apply(lambda x: age_groups(x)),
     'AnnualIncome':df['AnnualIncome'],
    'TravelInsurance':df['TravelInsurance']})

# Graph count plot of age groups (20s vs. 30s)
ax = sns.countplot(data=age_groups, x="Age_Groups", order=['20s', '30s'])

# add labels
for p, label in zip(ax.patches, age_groups.value_counts()):
    ax.annotate(label, (p.get_x()+0.37, p.get_height()+0.15))
plt.title("Age Groups (20s vs. 30s)")
plt.ylabel('Number of Customers')
plt.show()
```



4.1.5 Percentage of Travel Insurance Purchases on Various Features

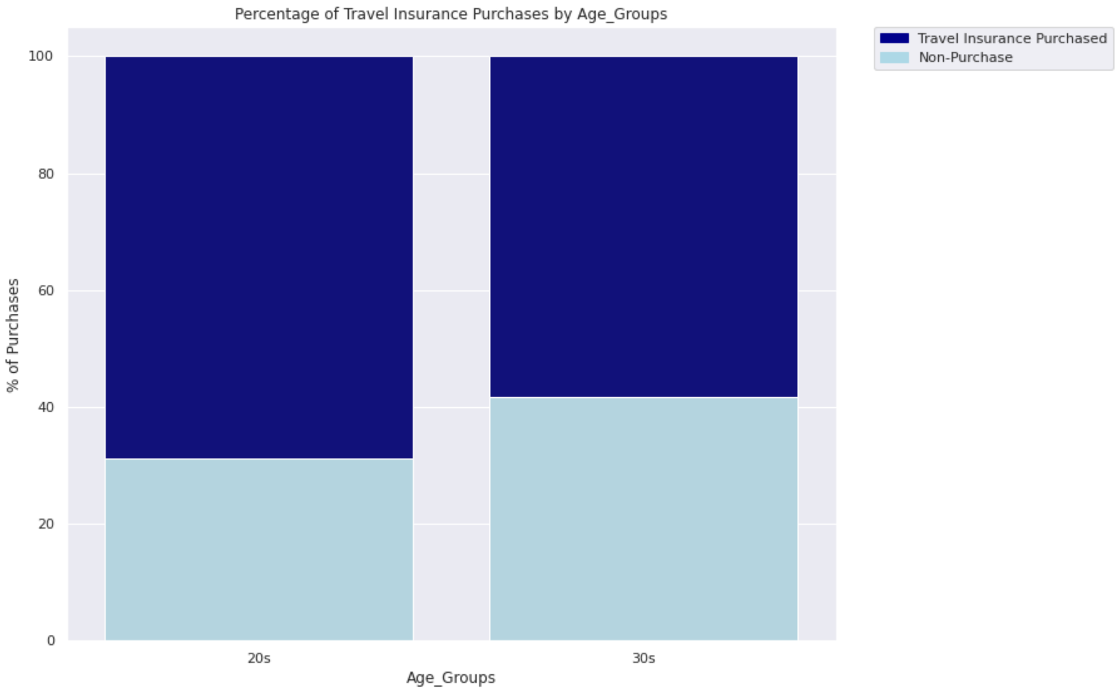
Python code:

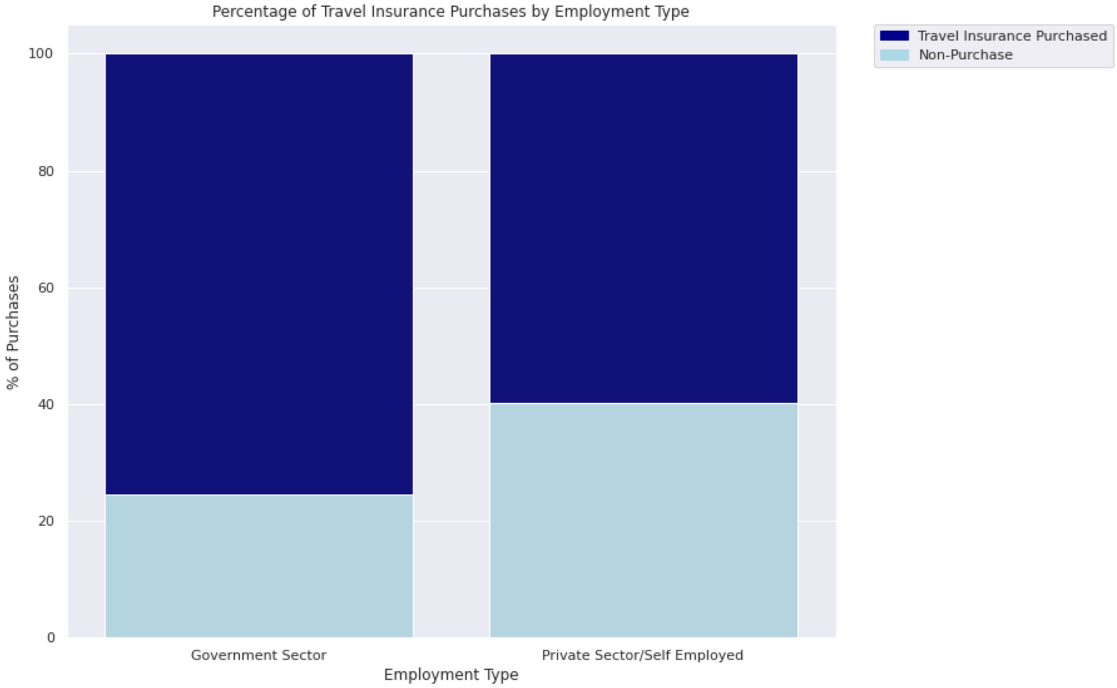
[9]:

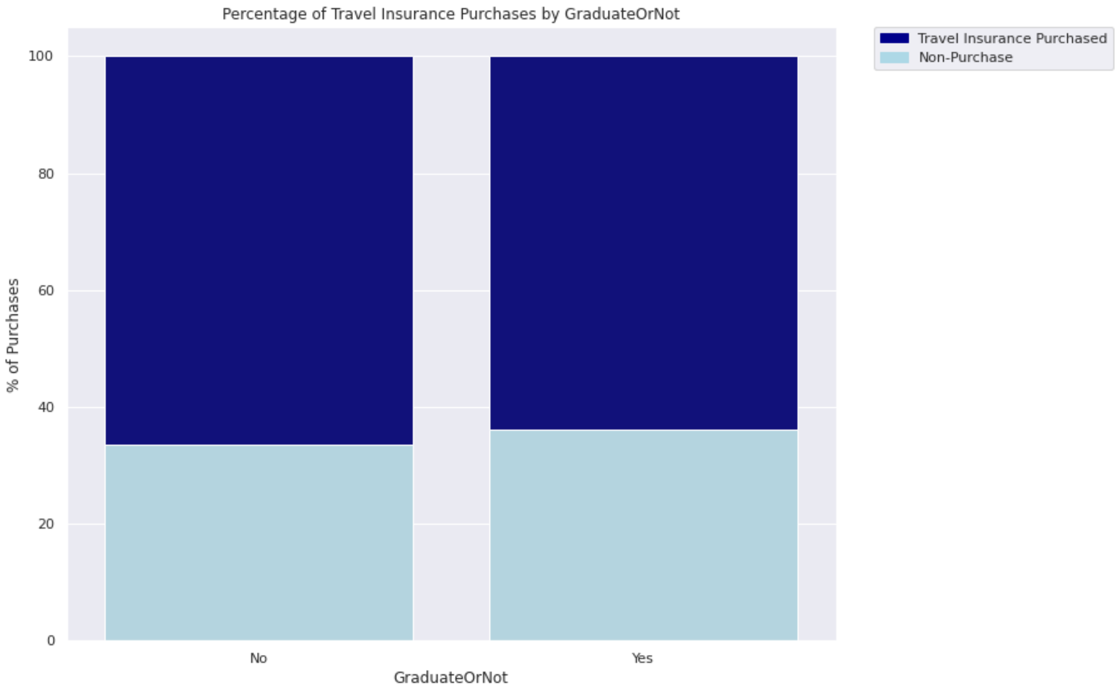
```
def make_stacked_barcharts(df, x):
    '''
    Takes in a data frame 'df' and a column 'x'
    and returns a stacked bar chart of the column
    with the percentage of purchases overlaid

    '''

    # Calculate total counts from both groups
    total = df.groupby(x)['TravelInsurance'].count().reset_index()

    # Calculate total counts from only purchases
    purchase = df[df.TravelInsurance == 1].groupby(
        x)['TravelInsurance'].sum().reset_index()

    # get percentages for purchases
    purchase['TravelInsurance'] = [
        i / j * 100
        for i, j in zip(purchase['TravelInsurance'], total['TravelInsurance'])
    ]

    # get percentages
    total['TravelInsurance'] = [
        i / j * 100
        for i, j in zip(total['TravelInsurance'], total['TravelInsurance'])
    ]

    # bar chart 1 -> top bars (group of 'TravelInsurance=0')
    bar1 = sns.barplot(x, y="TravelInsurance", data=total, color='darkblue')

    # bar chart 2 -> bottom bars (group of 'TravelInsurance=1')
    bar2 = sns.barplot(x,
                       y="TravelInsurance",
                       data=purchase,
                       color='lightblue')

    # add legend
    top_bar = mpatches.Patch(color='darkblue',
                             label='Travel Insurance Purchased')
    bottom_bar = mpatches.Patch(color='lightblue', label='Non-Purchase')
    plt.legend(handles=[top_bar, bottom_bar],
               bbox_to_anchor=(1.05, 1),
               loc=2,
               borderaxespad=0.)
    # Aesthetics
    plt.title("Percentage of Travel Insurance Purchases by " + x)
    plt.xlabel(x)
    plt.ylabel("% of Purchases")

    # Change ticks on x-axis for ChronicDiseases column
    if x == "ChronicDiseases":
        chronicdiease = [0, 1]
        labels = ['No', 'Yes']
        plt.xticks(chronicdiease, labels)

    # show the graph
    plt.show()
```

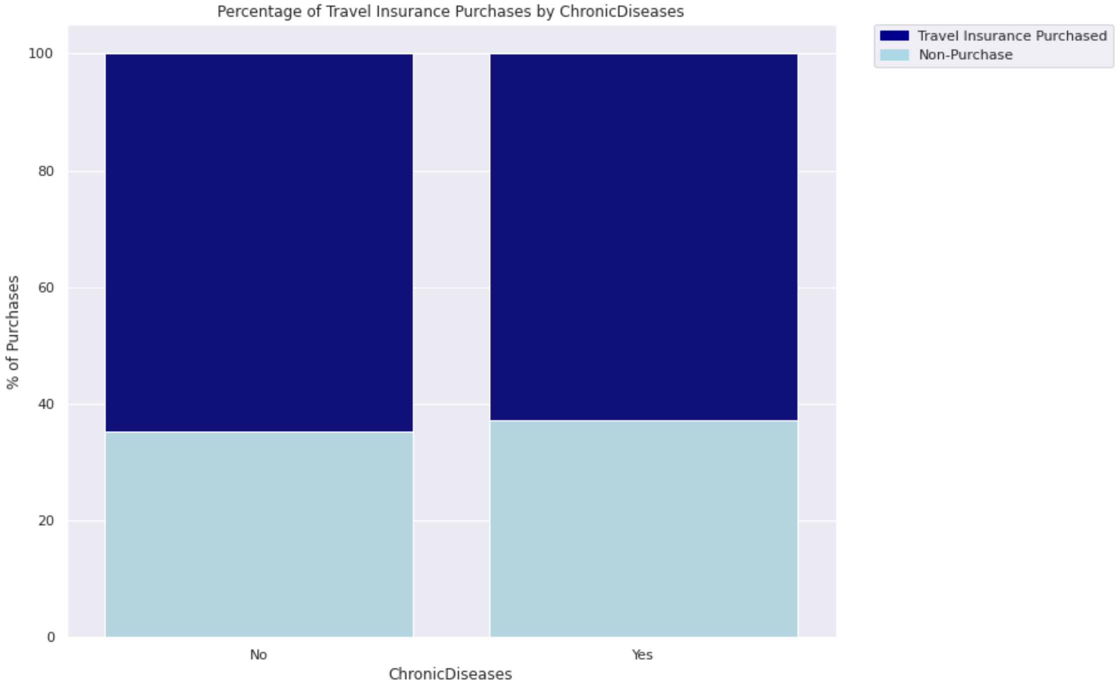
[10]:

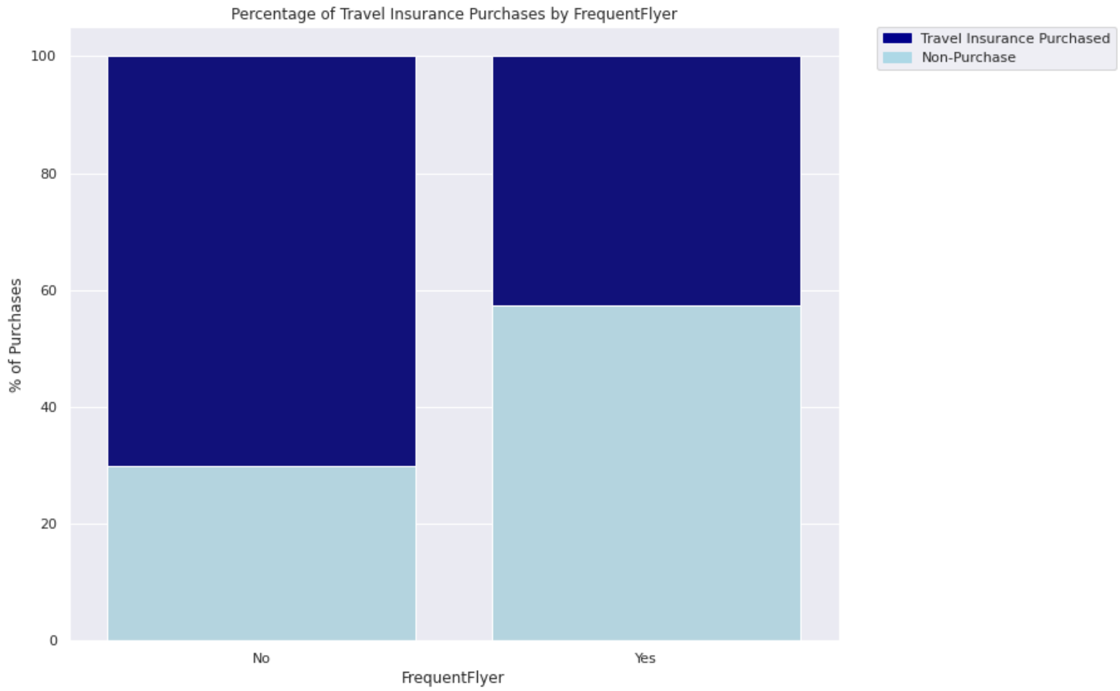
```
# Call stacked bar chart function
make_stacked_barcharts(age_groups, 'Age_Groups')

# Make stacked bar charts on various columns
plot_columns = [
    'Employment Type', 'GraduateOrNot', 'ChronicDiseases', 'FrequentFlyer',
    'EverTravelledAbroad'
]

# Plot each column
for i in plot_columns:
    make_stacked_barcharts(df, i)
```

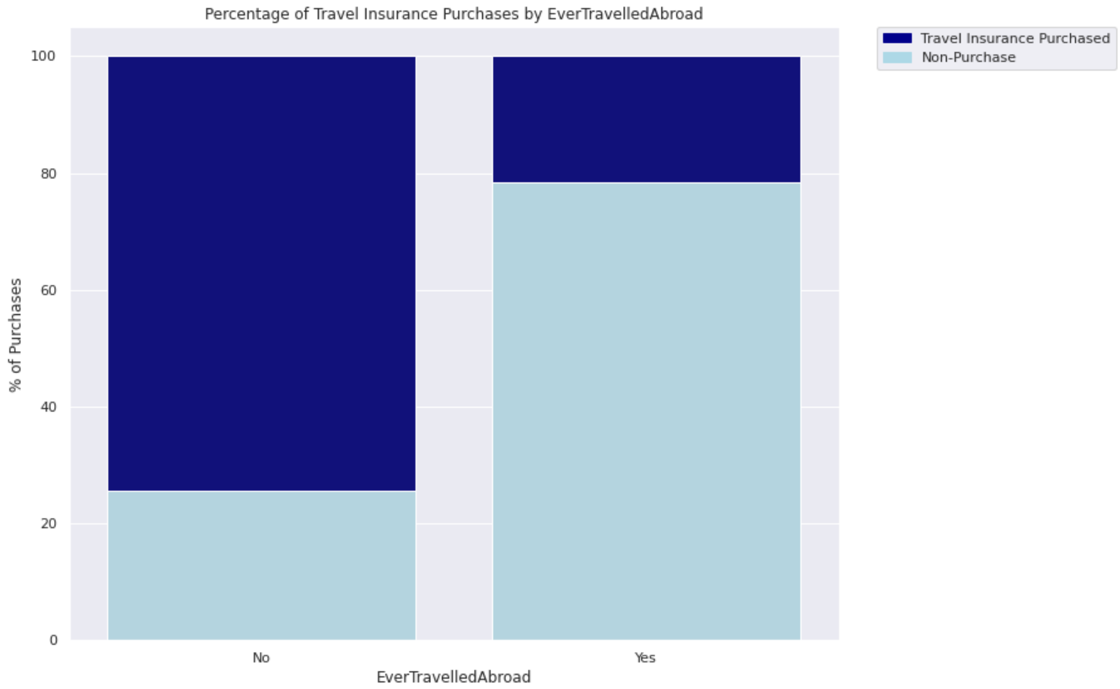

Percentage of Travel Insurance Purchases by Age_Groups
Travel Insurance Purchased
Non-Purchase
% of Purchases
0
20
40
60
80
100
20s
30s
Age_Groups


Percentage of Travel Insurance Purchases by Employment Type
Travel Insurance Purchased
Non-Purchase
% of Purchases
0
20
40
60
80
100
Government Sector
Private Sector/Self Employed
Employment Type


Percentage of Travel Insurance Purchases by GraduateOrNot
Travel Insurance Purchased
Non-Purchase
% of Purchases
0
20
40
60
80
100
No
Yes
GraduateOrNot


Percentage of Travel Insurance Purchases by ChronicDiseases
Travel Insurance Purchased
Non-Purchase
% of Purchases
0
20
40
60
80
100
No
Yes
ChronicDiseases





4.2 Side-by-side Box-plots between Annual Income and Different Attributes

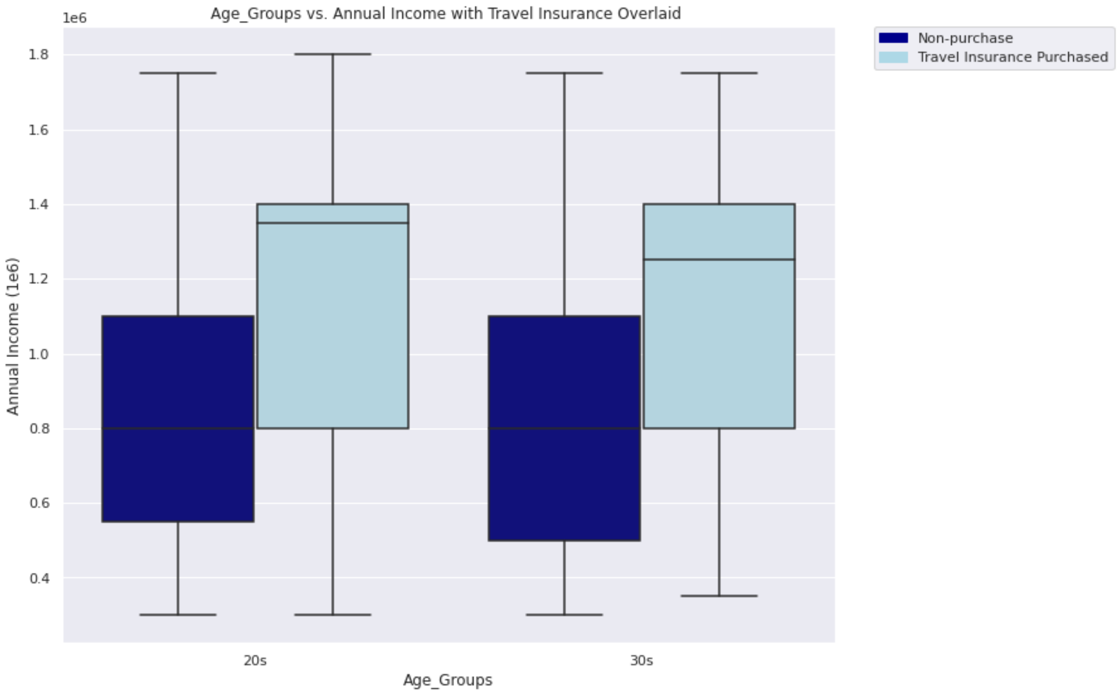
Python code:

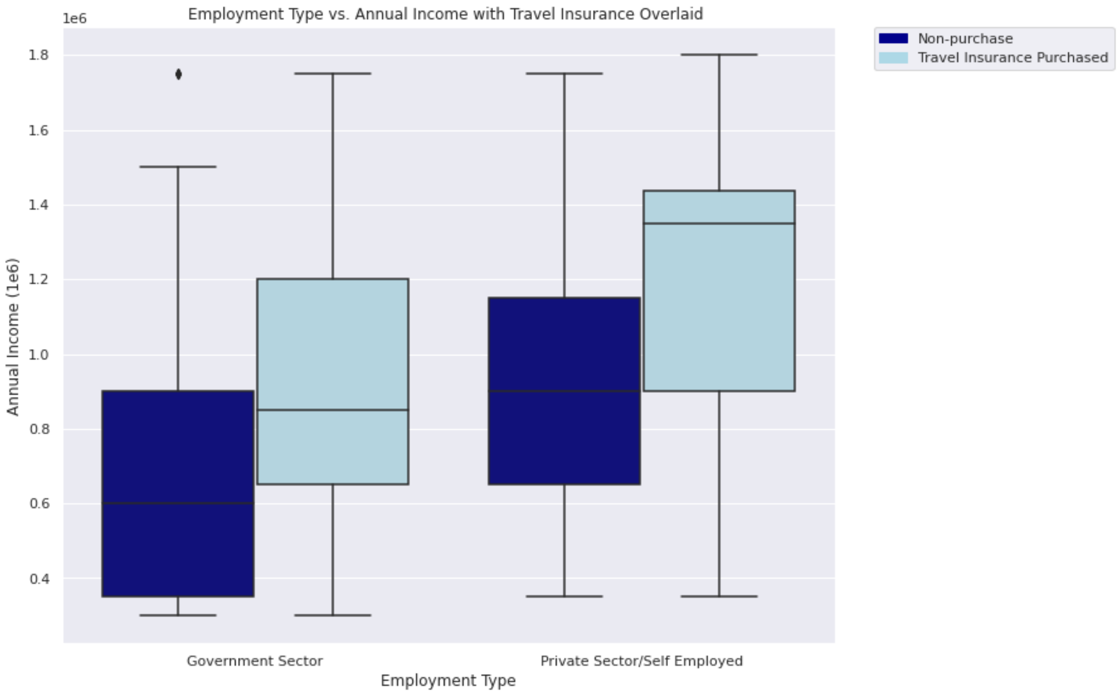
[11]:

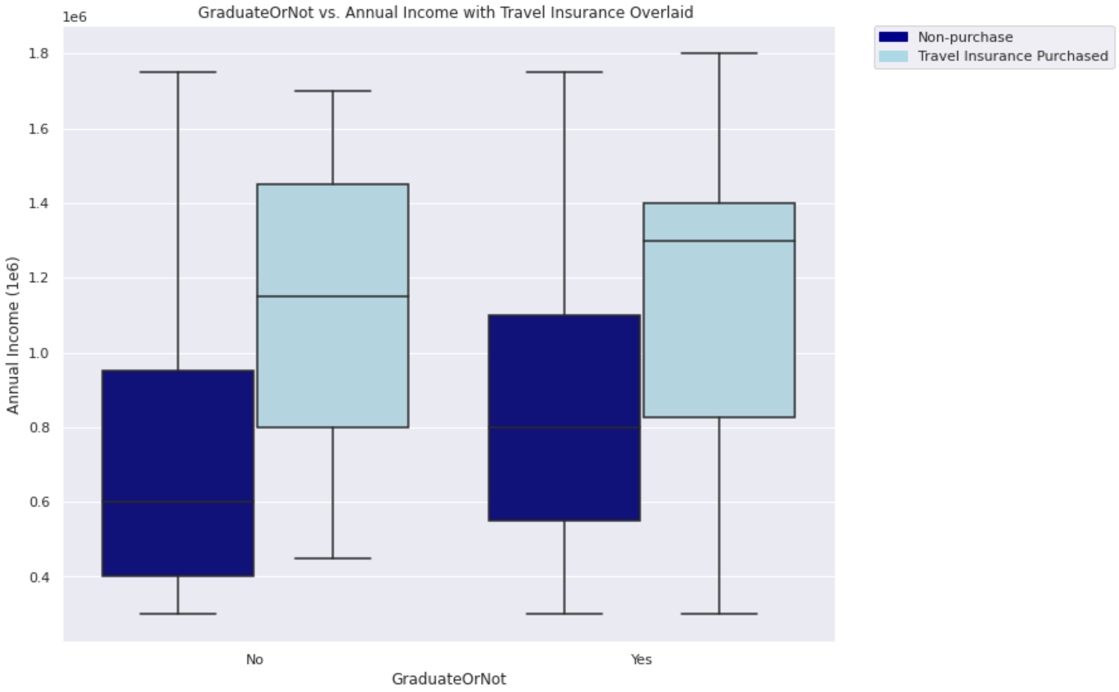
```
def make_boxplots(df, x):
    '''
    Takes in 'x' as a column from data frame 'df'
    and returns a side by side box-plot of
    x on the x-axis and AnnualIncome on the y-axis
    seperated by different colors noted by TravelInsurance

    '''

    # Palatte to color the target variable
    palatte = {0: "darkblue", 1: "lightblue"}

    # Change x-axis labels if age_groups or GraduatedOrNot
    order = None
    if x == "Age_Groups":
        order = ["20s", "30s"]
    if x == "GraduateOrNot":
        order = ["No", "Yes"]

    # Boxplot
    sns.boxplot(x=x,
                y="AnnualIncome",
                hue="TravelInsurance",
                data=df,
                order=order,
                palette=palatte)

    # Legend properties
    top_bar = mpatches.Patch(color='darkblue', label='Non-purchase')
    bottom_bar = mpatches.Patch(color='lightblue',
                                label='Travel Insurance Purchased')
    plt.legend(handles=[top_bar, bottom_bar],
               bbox_to_anchor=(1.05, 1),
               loc=2,
               borderaxespad=0.)

    # Graph Properties
    plt.title(x + " vs. Annual Income with Travel Insurance Overlaid ")
    plt.xlabel(x)
    plt.ylabel("Annual Income (1e6)")

    # Change ticks on x-axis for ChronicDiseases column
    if x == "ChronicDiseases":
        chronicdiease = [0, 1]
        labels = ['No', 'Yes']
        plt.xticks(chronicdiease, labels)
    # show the graph
    plt.show()
```

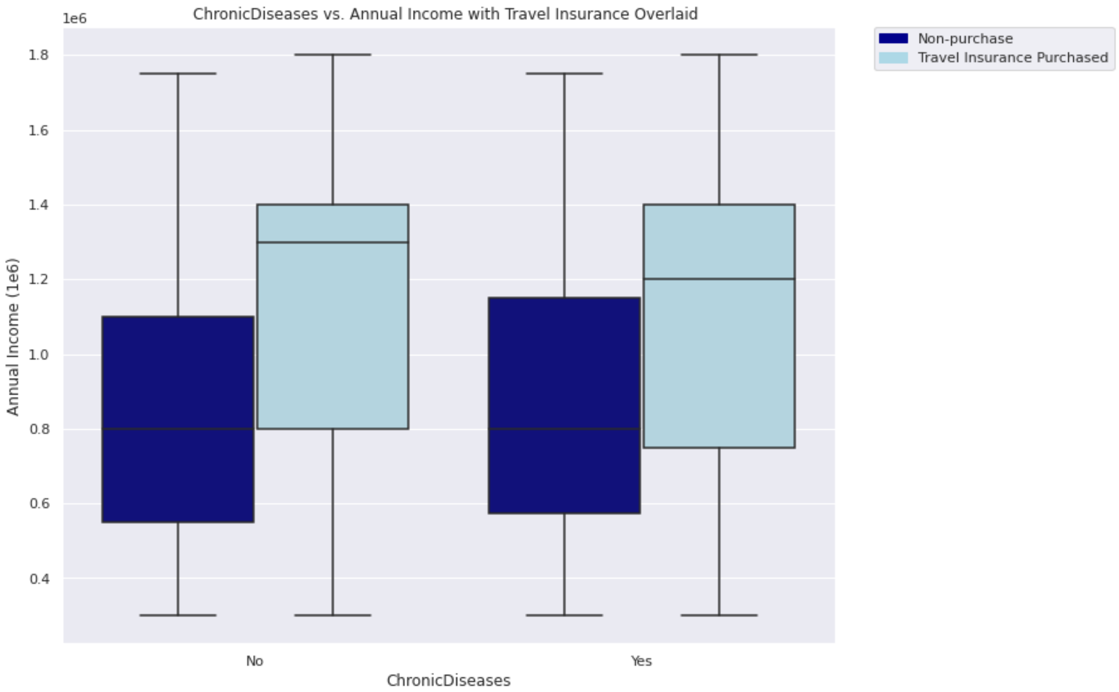
[12]:

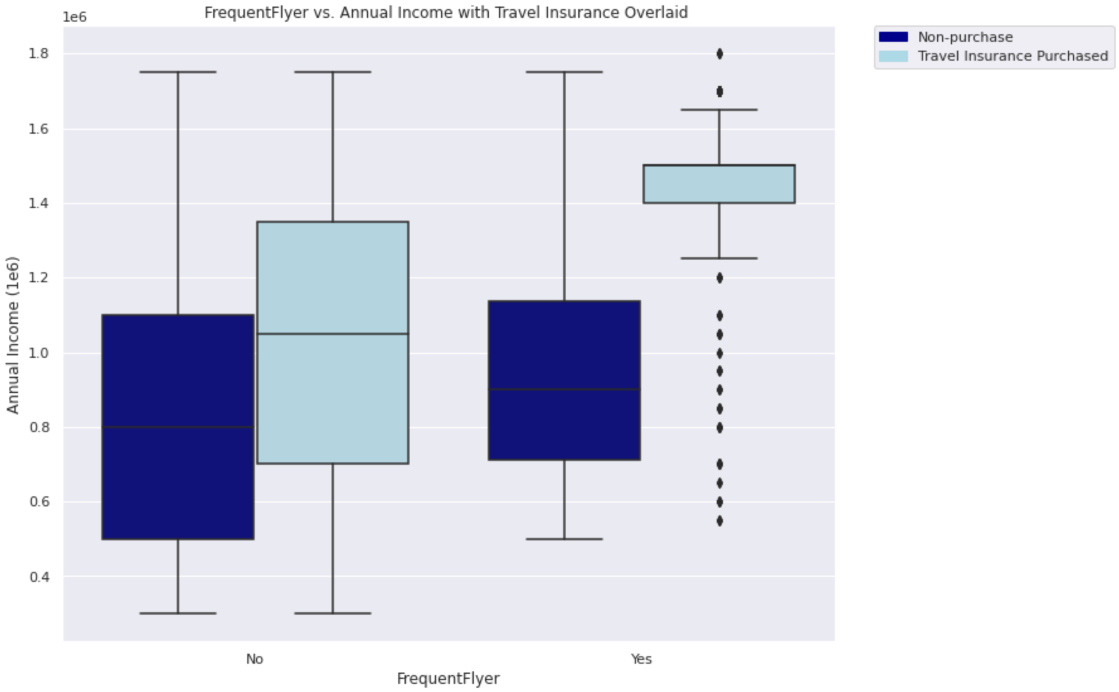
```
# call stacked bar chart function
make_boxplots(age_groups, 'Age_Groups')

# call boxplot function on various columns
plot_columns = ['Employment Type', 'GraduateOrNot','ChronicDiseases',␣
 ↪'FrequentFlyer',
       'EverTravelledAbroad']

# plot each column
for i in plot_columns:
    # boxplot function
    make_boxplots(df, i)
```

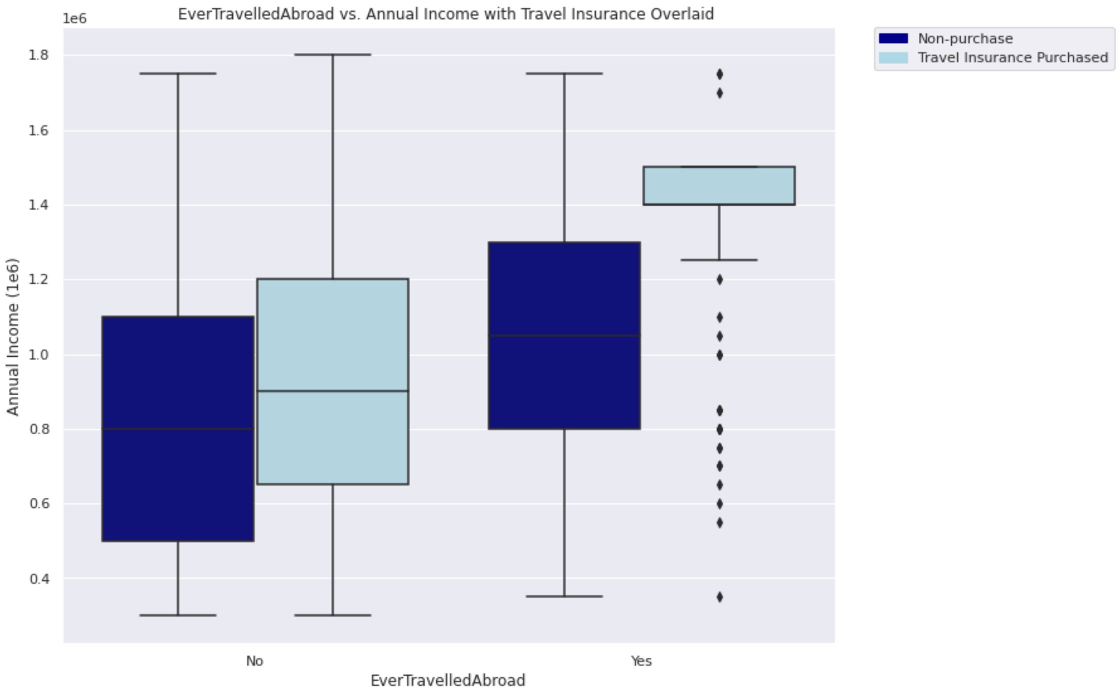



Employment Type vs. Annual Income with Travel Insurance Overlaid
1e6
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
Annual Income (1e6)
Government Sector
Private Sector/Self Employed
Employment Type
Non-purchase
Travel Insurance Purchased


GraduateOrNot vs. Annual Income with Travel Insurance Overlaid
1e6
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
Annual Income (1e6)
No
Yes
GraduateOrNot
Non-purchase
Travel Insurance Purchased


ChronicDiseases vs. Annual Income with Travel Insurance Overlaid
1e6
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
Annual Income (1e6)
No
Yes
ChronicDiseases
Non-purchase
Travel Insurance Purchased


FrequentFlyer vs. Annual Income with Travel Insurance Overlaid
1e6
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
Annual Income (1e6)
No
Yes
FrequentFlyer
Non-purchase
Travel Insurance Purchased



---

5 Data Wrangling and Pre-Processing

- Handling of missing values, outliers, correlated features, etc.

Python code:

[ ]:

---

6 Data splitting

- Training, validation, and test sets

Python code:

[ ]:

---

7 Model building strategies

- Describing main research questions and appropriate analytics methods

Python code:

```
[ ]:
```

---

8 Model performance and hyper-parameter tuning

- Model tuning, comparison, and evaluations

Python code:

```
[ ]:
```

---

9 Results and final model selection

- Performance measures on test Set

Python code:

```
[ ]:
```

---

10 Discussion and conclusion

- Address the problem statement and suggestions that could go beyond the scope of the course

```
[ ]:
```