databricks Caravan_EDA_MA755

Caravan EDA - MA755

Date: 02/27/2018
Author: MIMOZA MARKO (marko_mimo@bentley.edu), SNEHA ARORA (arora_sneh@bentley.edu), TAUSEEF AHMAD (ahmad_taus@bentley.edu)

Objective

The purpose of this notebook is to explain characteristics of customers. The analyses start with exploring the dataset and selecting the 10 most important variables in explaining the response variable CARAVAN. In order to perform the dimensionality reduction we apply the random forest classifier. After acquiring the important variables, we explore the single and multiple variable summaries to find patterns.

1. Dataset Description

The caravan insurance dataset supplied by the Dutch data mining company Sentient Machine Research and is based on a real-world business problem. The purpose of the project is to have a clear insight to why customers have a caravan policy and how these customers are different from the others.

The dataset consists of 87 attributes and 9822 observations. It is further divided into a training set (5822 observations) and a test set (4000 observations). Out of the 86 attributes, 2 are categorical, 83 are numerical and 1 is the response variable (Caravan Insurance Purchased) which indicates whether the customer purchase a caravan insurance policy or not. This is an imbalanced dataset as the target variable Caravan Insurance Purchased has more 0’s i.e. the customers did not purchase the insurance policy as compared to 1’s i.e. the customers did purchase the insurance policy. Futhermore, this dataset is set up as groups and each observation represent a large sample size.

There are 86 variables, containing sociodemographic data (variables 1-43) and product ownership data (variables 44-86). The sociodemographic data is derived from zip codes. All customers living in areas with the same zip code have the same sociodemographic attributes.

Each row of the dataset represents a group of customers. The following set of variables describe this group of customers:

- `MAANTHUI` : Number of houses 1 - 10 (in the group of customers)
- `MGEMOMV` : Avg size household 1 - 6 (in the group of customers)
- `MOSHOOFD` : Customer main type; see L2
- `MOSTYPE` : Customer Subtype; see L0
- `MGEMLEEF` : Avg age; see L1

See the documentation for the meaning of L0, L1 and L2.

https://www.kaggle.com/uciml/caravan-insurance-challenge/data (https://www.kaggle.com/uciml/caravan-insurance-challenge/data)

The remaining variables that start with "M" provide demographic data and contain integers between `0` and `9` that represent ranges of percentages with `0` representing 0% and `9` representing 100%. The other integers represent ranges of about 13% each. See the documentation for details.

The variables that start with "A" or "P" provide information about the customers in that postal code. They contain integers between `0` and `9` that represent ranges of counts for that variable. See the documentation for details.

The `CARAVAN` variable is the target variable. It is also a *count* variable (as above) recording the number of mobile home policies in that postal code.

2. Load Libraries

Load the required libraries and check version numbers.

```
import numpy             as np
import pandas            as pd
import matplotlib        as mpl
import matplotlib.pyplot as plt
import seaborn as sns
np.__version__, pd.__version__, mpl.__version__, sns.__version__
```

```
Out[2]: ('1.14.2', '0.18.1', '2.1.1', '0.8.1')
```

3. Read Dataset

First check that the files exists where we expect it.

```
%sh ls /dbfs/mnt/group-ma755/data
```

```
ais-test.csv
ais-train.csv
caravan-insurance-challenge.csv
```

Check that the file has a header and looks reasonable.

```
%sh head /dbfs/mnt/group-ma755/data/caravan-insurance-challenge.csv
```

```
ORIGIN,MOSTYPE,MAANTHUI,MGEMOMV,MGEMLEEF,MOSHOOFD,MGODRK,MGODPR,MGODOV,MGODGE,
MRELGE,MRELSA,MRELOV,MFALLEEN,MFGEKIND,MFWEKIND,MOPLHOOG,MOPLMIDD,MOPLLAAG,MBE
RHOOG,MBERZELF,MBERBOER,MBERMIDD,MBERARBG,MBERARBO,MSKA,MSKB1,MSKB2,MSKC,MSKD,
MHHUUR,MHKOOP,MAUT1,MAUT2,MAUT0,MZFONDS,MZPART,MINKM30,MINK3045,MINK4575,MINK7
512,MINK123M,MINKGEM,MKOOPKLA,PWAPART,PWABEDR,PWALAND,PPERSAUT,PBESAUT,PMOTSC
O,PVRAAUT,PAANHANG,PTRACTOR,PWERKT,PBROM,PLEVEN,PPERSONG,PGEZONG,PWAOREG,PBRAN
D,PZEILPL,PPLEZIER,PFIETS,PINBOED,PBYSTAND,AWAPART,AWABEDR,AWALAND,APERSAUT,AB
ESAUT,AMOTSCO,AVRAAUT,AAANHANG,ATRACTOR,AWERKT,ABROM,ALEVEN,APERSONG,AGEZONG,A
WAOREG,ABRAND,AZEILPL,APLEZIER,AFIETS,AINBOED,ABYSTAND,CARAVAN
train,33,1,3,2,8,0,5,1,3,7,0,2,1,2,6,1,2,7,1,0,1,2,5,2,1,1,2,6,1,1,8,8,0,1,8,
1,0,4,5,0,0,4,3,0,0,0,6,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,
0,0,0,0,0,1,0,0,0,0,0,0
train,37,1,2,2,8,1,4,1,4,6,2,2,0,4,5,0,5,4,0,0,0,5,0,4,0,2,3,5,0,2,7,7,1,2,6,
3,2,0,5,2,0,5,4,2,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,1,0,0,0,0,0,0
train,37,1,2,2,8,0,4,2,4,3,2,4,4,4,2,0,5,4,0,0,0,7,0,2,0,5,0,4,0,7,2,7,0,2,9,
0,4,5,0,0,0,3,4,2,0,0,6,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,
0,0,0,0,0,1,0,0,0,0,0,0
train,9,1,3,3,3,2,3,2,4,5,2,2,2,3,4,3,4,2,4,0,0,3,1,2,3,2,1,4,0,5,4,9,0,0,7,2,
1,5,3,0,0,4,4,0,0,0,6,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,
0,0,0,0,1,0,0,0,0,0,0
```

```
caravan_df = pd.read_csv('/dbfs/mnt/group-ma755/data/caravan-insurance-
challenge.csv')
caravan_df.head()
```

```
Out[5]:
  ORIGIN  MOSTYPE  MAANTHUI  MGEMOMV  MGEMLEEF  MOSHOOFD  MGODRK  MGODPR  \
0  train       33         1        3         2         8       0       5
1  train       37         1        2         2         8       1       4
2  train       37         1        2         2         8       0       4
3  train        9         1        3         3         3       2       3
4  train       40         1        4         2        10       1       4

   MGODOV  MGODGE   ...     APERSONG  AGEZONG  AWAOREG  ABRAND  AZEILPL  \
0       1       3   ...            0        0        0       1        0
1       1       4   ...            0        0        0       1        0
2       2       4   ...            0        0        0       1        0
3       2       4   ...            0        0        0       1        0
4       1       4   ...            0        0        0       1        0

   APLEZIER  AFIETS  AINBOED  ABYSTAND  CARAVAN
0         0       0        0         0        0
1         0       0        0         0        0
2         0       0        0         0        0
3         0       0        0         0        0
4         0       0        0         0        0
```

Check Dataset

```
caravan_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9822 entries, 0 to 9821
Data columns (total 87 columns):
ORIGIN      9822 non-null object
MOSTYPE     9822 non-null int64
MAANTHUI    9822 non-null int64
MGEMOMV     9822 non-null int64
MGEMLEEF    9822 non-null int64
MOSHOOFD    9822 non-null int64
MGODRK      9822 non-null int64
MGODPR      9822 non-null int64
MGODOV      9822 non-null int64
MGODGE      9822 non-null int64
MRELGE      9822 non-null int64
```

```
MRELSA      9822 non-null int64
MRELOV      9822 non-null int64
MFALLEEN    9822 non-null int64
MFGEKIND    9822 non-null int64
MFWEKIND    9822 non-null int64
MOPLHOOG    9822 non-null int64
```

```
caravan_df.columns
```

```
Out[7]:
Index(['ORIGIN', 'MOSTYPE', 'MAANTHUI', 'MGEMOMV', 'MGEMLEEF', 'MOSHOOFD',
       'MGODRK', 'MGODPR', 'MGODOV', 'MGODGE', 'MRELGE', 'MRELSA', 'MRELOV',
       'MFALLEEN', 'MFGEKIND', 'MFWEKIND', 'MOPLHOOG', 'MOPLMIDD', 'MOPLLAAG',
       'MBERHOOG', 'MBERZELF', 'MBERBOER', 'MBERMIDD', 'MBERARBG', 'MBERARBO',
       'MSKA', 'MSKB1', 'MSKB2', 'MSKC', 'MSKD', 'MHHUUR', 'MHKOOP', 'MAUT1',
       'MAUT2', 'MAUT0', 'MZFONDS', 'MZPART', 'MINKM30', 'MINK3045',
       'MINK4575', 'MINK7512', 'MINK123M', 'MINKGEM', 'MKOOPKLA', 'PWAPART',
       'PWABEDR', 'PWALAND', 'PPERSAUT', 'PBESAUT', 'PMOTSCO', 'PVRAAUT',
       'PAANHANG', 'PTRACTOR', 'PWERKT', 'PBROM', 'PLEVEN', 'PPERSONG',
       'PGEZONG', 'PWAOREG', 'PBRAND', 'PZEILPL', 'PPLEZIER', 'PFIETS',
       'PINBOED', 'PBYSTAND', 'AWAPART', 'AWABEDR', 'AWALAND', 'APERSAUT',
       'ABESAUT', 'AMOTSCO', 'AVRAAUT', 'AAANHANG', 'ATRACTOR', 'AWERKT',
       'ABROM', 'ALEVEN', 'APERSONG', 'AGEZONG', 'AWAOREG', 'ABRAND',
       'AZEILPL', 'APLEZIER', 'AFIETS', 'AINBOED', 'ABYSTAND', 'CARAVAN'],
      dtype='object')
```

There are 87 variables and 9,822 observations in this dataframe.

```
caravan_df.shape
```

```
Out[8]: (9822, 87)
```

The `'Origin'` variable contains information if the observation is train or test. 5,822 of the observations are train and 4,000 are tests.

```
sum(caravan_df['ORIGIN']== 'train'), sum(caravan_df['ORIGIN']== 'test')
```

```
Out[9]: (5822, 4000)
```

4. Variable Selection

As a first step in this section we choose which variables to keep for analyzing. We first check the correlation of the numerical variables, then from the ones that are highly correlated we only keep one variable per each pair. We apply random forest classifier to check the variable importance.

```
numerical_columns = [name for name in list(caravan_df.columns) if name not in
['ORIGIN', 'CARAVAN', 'MOSTYPE', 'MOSHOOFD']]
```

Apply the `corr()` method to obtain the correlation table for the numerical columns of the `caravan_df`. The output is a pandas dataframe that we store in `correlation_df`.

```
correlation_df = caravan_df[numerical_columns].corr(method="pearson") # apply
the corr() method to numerical columns
correlation_df # pandas dataframe
```

```
Out[11]:
          MAANTHUI   MGEMOMV  MGEMLEEF    MGODRK    MGODPR    MGODOV  \
MAANTHUI  1.000000 -0.004315  0.052448  0.004304 -0.031296  0.020160
MGEMOMV  -0.004315  1.000000 -0.339942  0.009383  0.053455 -0.113296
MGEMLEEF  0.052448 -0.339942  1.000000 -0.038454  0.091775  0.058412
MGODRK    0.004304  0.009383 -0.038454  1.000000 -0.365998  0.022942
MGODPR   -0.031296  0.053455  0.091775 -0.365998  1.000000 -0.318426
MGODOV    0.020160 -0.113296  0.058412  0.022942 -0.318426  1.000000
MGODGE    0.018018 -0.010404 -0.114922 -0.079110 -0.743722 -0.130609
MRELGE    0.006801  0.531272 -0.058155 -0.021026  0.138014 -0.134896
MRELSA   -0.029157 -0.178395 -0.293537  0.115748 -0.215578  0.113309
MRELOV    0.000420 -0.496816  0.212892 -0.013744 -0.068782  0.114728
MFALLEEN  0.046156 -0.653864  0.245442  0.013623 -0.108196  0.131500
MFGEKIND -0.075771 -0.321324  0.212900  0.004721  0.044838  0.022794
MFWEKIND  0.024596  0.791445 -0.371329 -0.017466  0.034227 -0.112366
MOPLHOOG  0.005784  0.017234 -0.030063  0.244283 -0.099757  0.004198
MOPLMIDD -0.054934  0.039817 -0.219489  0.136335 -0.010965  0.019623
MOPLLAAG  0.034323 -0.045710  0.176600 -0.246624  0.058878 -0.012998
MBERHOOG -0.004755  0.030658  0.135751  0.223981 -0.015493 -0.046798
MBERZELF  0.034324  0.047274  0.042961  0.064208  0.039569  0.012362
MBERBOER -0.023410  0.102948  0.087396 -0.109850  0.115508 -0.045008
```

Filter the `correlation_df` to find those variables whose absolute correlation value is greater than .7. The output is a pandas dataframe.

```
correlation_df[abs(correlation_df) > .7 ]
```

```
Out[12]:
          MAANTHUI   MGEMOMV  MGEMLEEF  MGODRK    MGODPR  MGODOV    MGODGE  \
MAANTHUI       1.0       NaN       NaN     NaN       NaN     NaN       NaN
MGEMOMV        NaN  1.000000       NaN     NaN       NaN     NaN       NaN
MGEMLEEF       NaN       NaN       1.0     NaN       NaN     NaN       NaN
MGODRK         NaN       NaN       NaN     1.0       NaN     NaN       NaN
MGODPR         NaN       NaN       NaN     NaN  1.000000     NaN -0.743722
MGODOV         NaN       NaN       NaN     NaN       NaN     1.0       NaN
MGODGE         NaN       NaN       NaN     NaN -0.743722     NaN  1.000000
MRELGE         NaN       NaN       NaN     NaN       NaN     NaN       NaN
MRELSA         NaN       NaN       NaN     NaN       NaN     NaN       NaN
MRELOV         NaN       NaN       NaN     NaN       NaN     NaN       NaN
MFALLEEN       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MFGEKIND       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MFWEKIND       NaN  0.791445       NaN     NaN       NaN     NaN       NaN
MOPLHOOG       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MOPLMIDD       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MOPLLAAG       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MBERHOOG       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MBERZELF       NaN       NaN       NaN     NaN       NaN     NaN       NaN
MBERBOER       NaN       NaN       NaN     NaN       NaN     NaN       NaN
```

Applying the `stack()` method to the to the above correlation dataframe will return all the pairs associated with the correlation magnitude. The default of the method is to remove the NaN values. Therefore, we get only the pairs with values. The output of `stack()` here is a pandas series.

```
# return pd series with the pairs of variables and their correlation magnitude
that satifies the filter.
correlation_df[abs(correlation_df) > .7 ].stack()
```

```
Out[13]:
MAANTHUI  MAANTHUI    1.000000
MGEMOMV   MGEMOMV     1.000000
          MFWEKIND    0.791445
MGEMLEEF  MGEMLEEF    1.000000
MGODRK    MGODRK      1.000000
MGODPR    MGODPR      1.000000
          MGODGE     -0.743722
MGODOV    MGODOV      1.000000
MGODGE    MGODPR     -0.743722
          MGODGE      1.000000
MRELGE    MRELGE      1.000000
```

```
          MRELOV     -0.883251
MRELSA    MRELSA      1.000000
MRELOV    MRELGE     -0.883251
          MRELOV      1.000000
          MFALLEEN    0.741809
MFALLEEN  MRELOV      0.741809
          MFALLEEN    1.000000
MFGEKIND  MFGEKIND    1.000000
MFWEKIND  MGEMOMV     0.791445
```

The code below returns a lists of the indices with correlation magnitude higher than 0.7. The `index()` method returns the indices, which in this case are tuples. Check the above series for reference.

```
correlated_pairs=list(correlation_df[abs(correlation_df) > .7 ].stack().index)
correlated_pairs
```

```
Out[14]:
[('MAANTHUI', 'MAANTHUI'),
 ('MGEMOMV', 'MGEMOMV'),
 ('MGEMOMV', 'MFWEKIND'),
 ('MGEMLEEF', 'MGEMLEEF'),
 ('MGODRK', 'MGODRK'),
 ('MGODPR', 'MGODPR'),
 ('MGODPR', 'MGODGE'),
 ('MGODOV', 'MGODOV'),
 ('MGODGE', 'MGODPR'),
 ('MGODGE', 'MGODGE'),
 ('MRELGE', 'MRELGE'),
 ('MRELGE', 'MRELOV'),
 ('MRELSA', 'MRELSA'),
 ('MRELOV', 'MRELGE'),
 ('MRELOV', 'MRELOV'),
 ('MRELOV', 'MFALLEEN'),
 ('MFALLEEN', 'MRELOV'),
 ('MFALLEEN', 'MFALLEEN'),
 ('MFGEKIND', 'MFGEKIND'),
 ('MFWEKIND', 'MGEMOMV'),
```

Filter the `correlated_pairs` to exclude the pairs with the same variable, since it is the correlation of that variable and itself.

```
correlated_unique_pairs = [tuple for tuple in correlated_pairs if (tuple[0] !=
tuple[1])]
correlated_unique_pairs
```

```
Out[15]:
[('MGEMOMV', 'MFWEKIND'),
 ('MGODPR', 'MGODGE'),
 ('MGODGE', 'MGODPR'),
 ('MRELGE', 'MRELOV'),
 ('MRELOV', 'MRELGE'),
 ('MRELOV', 'MFALLEEN'),
 ('MFALLEEN', 'MRELOV'),
 ('MFWEKIND', 'MGEMOMV'),
 ('MOPLHOOG', 'MSKA'),
 ('MOPLMIDD', 'MOPLLAAG'),
 ('MOPLLAAG', 'MOPLMIDD'),
 ('MSKA', 'MOPLHOOG'),
 ('MHHUUR', 'MHKOOP'),
 ('MHKOOP', 'MHHUUR'),
 ('MAUT1', 'MAUT0'),
 ('MAUT0', 'MAUT1'),
 ('MZFONDS', 'MZPART'),
 ('MZPART', 'MZFONDS'),
 ('PWAPART', 'AWAPART'),
 ('PWABEDR', 'AWABEDR'),
```

The variables below has been eliminated because they were highly correlated with other variables. These variables do not provide incremental information about the response variable.

```
eliminated_variables =['MHKOOP', 'MZFONDS', 'PWALAND', 'PWAPART', 'PGEZONG',
'PBROM', 'PBYSTAND', 'PAANHANG', 'PWAOREG', 'PZEILPL', 'PFIETS',   'PMOTSCO',
'PTRACTOR',  'PWERKT',  'PPERSAUT', 'PWABEDR', 'PVRAAUT', 'PPERSONG',
'PBESAUT', 'PPLEZIER', 'MRELOV', 'PBRAND',  'PINBOED', 'PLEVEN', 'MFWEKIND',
'MGODGE', 'MFALLEEN', 'MOPLMIDD', 'MAUT1', 'MOPLHOOG']
len(eliminated_variables)
```

```
Out[16]: 30
```

Below is displayed the list of variables that has been chosen from each pair of highly correlated variables.

```
selected_variables = ['MHHUUR', 'MZPART', 'AWALAND', 'AWAPART', 'AGEZONG',
'ABROM', 'ABYSTAND', 'AAANHANG', 'AWAOREG', 'AZEILPL', 'AFIETS', 'AMOTSCO',
'ATRACTOR', 'AWERKT',  'APERSAUT',  'AWABEDR',  'AVRAAUT',  'APERSONG',
'ABESAUT', 'APLEZIER', 'MRELGE', 'ABRAND', 'AINBOED', 'ALEVEN', 'MGEMOMV',
'MGODPR', 'MFALLEEN', 'MOPLLAAG', 'MAUT0', 'MSKA']
len(selected_variables)
```

```
Out[17]: 30
```

The final list of variables for random forest classifier includes the selected numerical variables above, the numerical variables which were not highly correlated with any other variable and the categorical variables.

```
rand_forest_var = [ var for var in caravan_df.columns if var not in
eliminated_variables and var not in ['CARAVAN', 'ORIGIN']]
```

```
len(rand_forest_var)
```

```
Out[19]: 55
```

Divide the data into train and test using only the random forest variables.

```
x_train= caravan_df.loc[caravan_df['ORIGIN']=='train', rand_forest_var]
x_test = caravan_df.loc[caravan_df['ORIGIN']=='test', rand_forest_var]
```

Define the `y_train` to be used in random forest classifier.

```
y_train=caravan_df.loc[caravan_df['ORIGIN']=='train','CARAVAN']
```

We perform the random forest classifier on the selected variables 55 variables.

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=2, random_state=0) # max_depth the depth
of the tree, random_state is the random number generator
clf.fit(x_train, y_train)
RandomForestClassifier(bootstrap=True, criterion='gini') # bootsrap sampling is
done, gini because it is classification
```

```
Out[22]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

Display the feature importance.

```
clf.feature_importances_
```

```
Out[23]:
array([0.        , 0.        , 0.        , 0.        , 0.02680009,
       0.        , 0.0310786 , 0.        , 0.        , 0.        ,
       0.01221705, 0.0707111 , 0.        , 0.        , 0.        ,
       0.        , 0.01423333, 0.03382929, 0.        , 0.        ,
       0.        , 0.        , 0.01973396, 0.07491502, 0.        ,
       0.        , 0.        , 0.08627368, 0.        , 0.00770941,
       0.03507074, 0.        , 0.12319854, 0.09132489, 0.08523247,
       0.        , 0.        , 0.14391677, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.06428203, 0.        , 0.        , 0.        , 0.03573934,
       0.        , 0.0437337 , 0.        , 0.        , 0.        ])
```

Zip the columns with their corresponding feature importance and store it as a list of tuples in `variable_importance`.

```
variable_importances = list(zip(x_train.columns,clf.feature_importances_))
variable_importances
```

```
Out[24]:
[('MOSTYPE', 0.0),
 ('MAANTHUI', 0.0),
 ('MGEMOMV', 0.0),
 ('MGEMLEEF', 0.0),
 ('MOSHOOFD', 0.0268000858175244),
 ('MGODRK', 0.0),
 ('MGODPR', 0.03107859638287397),
 ('MGODOV', 0.0),
 ('MRELGE', 0.0),
 ('MRELSA', 0.0),
 ('MFGEKIND', 0.012217052498740881),
 ('MOPLLAAG', 0.0707110954272083),
 ('MBERHOOG', 0.0),
 ('MBERZELF', 0.0),
 ('MBERBOER', 0.0),
 ('MBERMIDD', 0.0),
 ('MBERARBG', 0.014233331315897685),
 ('MBERARBO', 0.03382929410894868),
 ('MSKA', 0.0),
 ('MSKB1', 0.0),
```

Sort the above list according to the second item of the tuples which is the importance magnitude.

```
sorted_var_importances = sorted(variable_importances, key=lambda x: x[1],
reverse = True)
sorted_var_importances
```

```
Out[25]:
[('APERSAUT', 0.14391676953794413),
 ('MINKGEM', 0.12319853835130132),
 ('MKOOPKLA', 0.09132488981010675),
 ('MINKM30', 0.08627368409867325),
 ('AWAPART', 0.08523247123055566),
 ('MHHUUR', 0.07491501678054516),
 ('MOPLLAAG', 0.0707110954272083),
 ('ALEVEN', 0.06428202924369271),
 ('APLEZIER', 0.04373369830478013),
 ('ABRAND', 0.03573934000016023),
 ('MINK7512', 0.03507074026432441),
 ('MBERARBO', 0.03382929410894868),
 ('MGODPR', 0.03107859638287397),
 ('MOSHOOFD', 0.0268000858175244),
 ('MSKD', 0.01973395959074747),
 ('MBERARBG', 0.014233331315897685),
 ('MFGEKIND', 0.012217052498740881),
 ('MINK4575', 0.007709407235974831),
 ('MOSTYPE', 0.0),
 ('MAANTHUI', 0.0),
```

Top 10 most important variables according to the random forest result. The last one is the response CARAVAN.

```
most_imp_var= ['APERSAUT', 'MINKGEM', 'MKOOPKLA','MINKM30','AWAPART', 'MHHUUR',
'MOPLLAAG', 'ALEVEN', 'APLEZIER', 'ABRAND', 'CARAVAN']
```

Create the reduced dimension dataset to contain only the selected variables.

```
red_caravan_df = caravan_df[most_imp_var]
```

Display the structure of the new dataset.

```
red_caravan_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9822 entries, 0 to 9821
Data columns (total 11 columns):
APERSAUT    9822 non-null int64
MINKGEM     9822 non-null int64
MKOOPKLA    9822 non-null int64
MINKM30     9822 non-null int64
AWAPART     9822 non-null int64
MHHUUR      9822 non-null int64
MOPLLAAG    9822 non-null int64
ALEVEN      9822 non-null int64
APLEZIER    9822 non-null int64
ABRAND      9822 non-null int64
CARAVAN     9822 non-null int64
dtypes: int64(11)
memory usage: 844.2 KB
```

5. Explore Single Variables

5.1 CARAVAN - Caravan Insurance Policy

Number of mobile home policies: 0 (no policy) or 1 (having policy)

```
grouped = red_caravan_df.groupby('CARAVAN')
grouped['CARAVAN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort_values(by='percentage', ascending =False)
```

```
Out[29]:
         count  percentage
CARAVAN
```

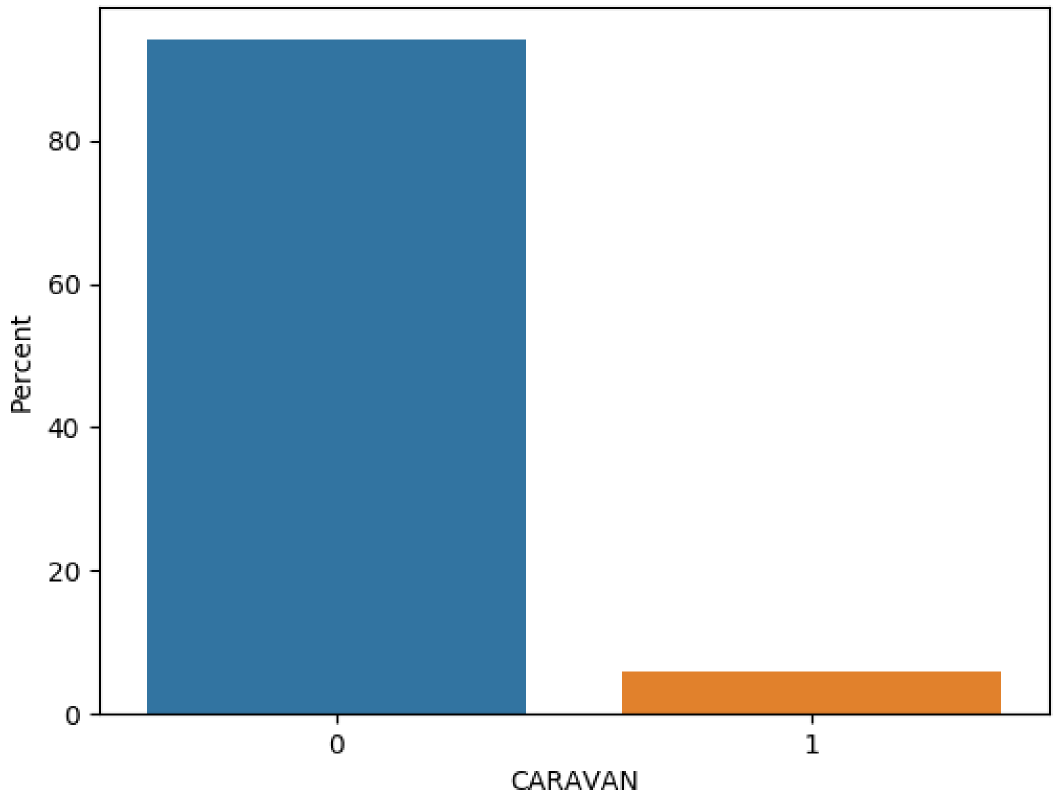
```
0         9236    0.940338
1          586    0.059662
```

6% of all the zipcodes bought caravan insurance policy.

```
caravan_plot = sns.barplot(x='CARAVAN', y='CARAVAN', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(ylabel="Percent")

Out[30]: [Text(0,0.5,'Percent')]

display()
```



This graph visualizes the percentages of the above table. 94% of the zip codes did not purchase the insurance.

5.2 APERSAUT - Number of Car policies

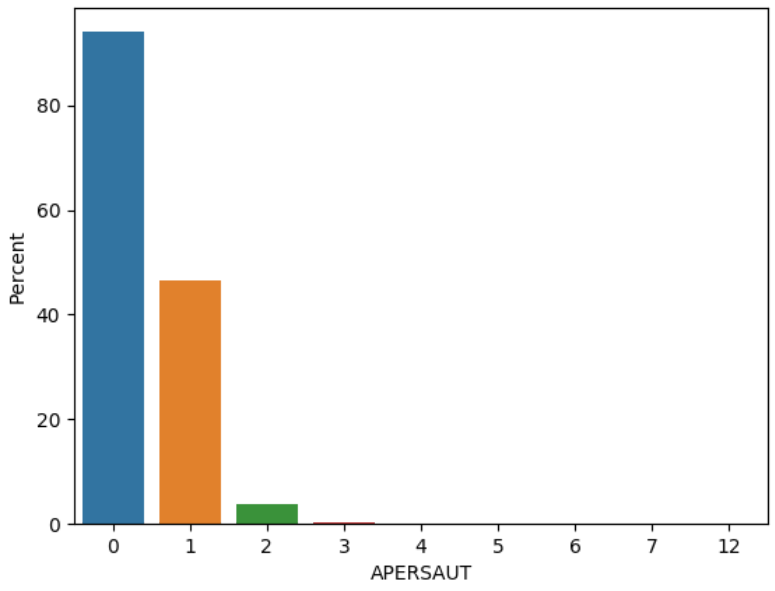
```
grouped = red_caravan_df.groupby('APERSAUT')
grouped['APERSAUT'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
/tmp/1522871921326-0/PythonShell.py:3: FutureWarning: sort(columns=....) is de
precated, use sort_values(by=.....)

Out[32]:
          count  percentage
APERSAUT
0          4825    0.491244
1          4580    0.466300
2           384    0.039096
3            21    0.002138
4             8    0.000814
5             1    0.000102
6             1    0.000102
7             1    0.000102
12            1    0.000102
```

The above result shows that most of the zipcodes are contentrated in the low levels of car insurance. 50% of them do not have any car insurance, 47% have 1-49 car insurances and 4% have 50-99 car insurances.

```
caravan_plot = sns.barplot(x='APERSAUT', y='APERSAUT', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the above result. We can see clearly how all of the obseravtions fall into the low levels of car insurance policies.

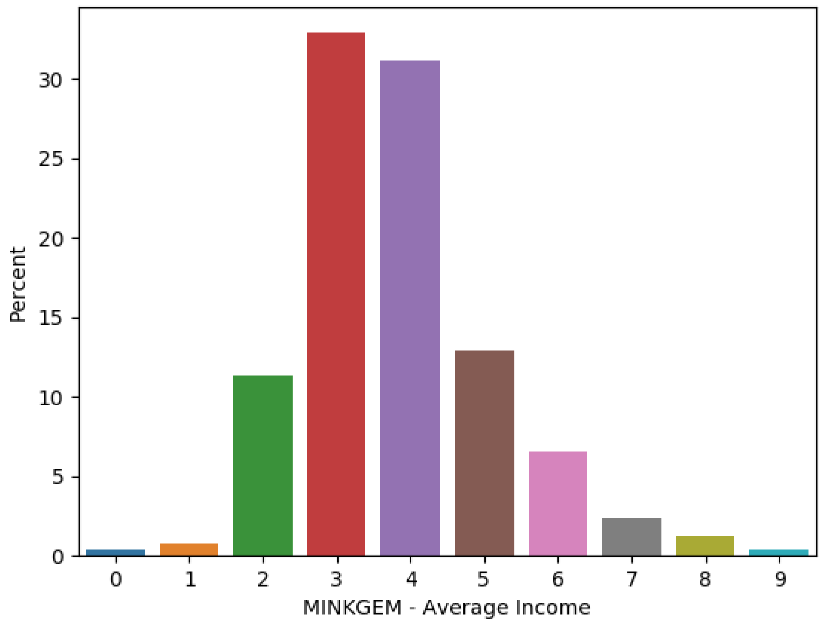
5.3 MINKGEM - Average Income

```
grouped = red_caravan_df.groupby('MINKGEM')
grouped['MINKGEM'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[34]:
         count  percentage
MINKGEM
3         3232    0.329057
4         3063    0.311851
5         1268    0.129098
2         1110    0.113012
6          646    0.065771
7          228    0.023213
8          121    0.012319
1           78    0.007941
0           38    0.003869
9           38    0.003869
```

The group by result shows that 33% of the zip codes have 24-36% average income and 32% of the zip codes have 37-49% average income. Our data mostly (about 64%) contains areas with 24-49% average income.

```
caravan_plot = sns.barplot(x='MINKGEM', y='MINKGEM', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel="MINKGEM - Average Income" ,ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the average income result. We can see clearly how all of the observations fall into the middle levels average income.

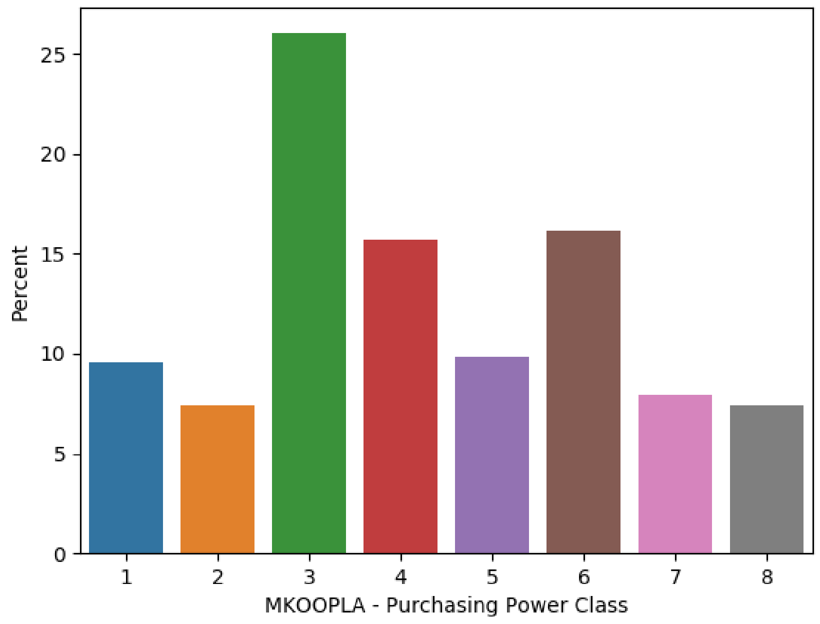
5.4 MKOOPKLA - Purchasing Power Class

```
grouped = red_caravan_df.groupby('MKOOPKLA')
grouped['MKOOPKLA'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[107]:
          count  percentage
MKOOPKLA
3          2556    0.260232
6          1587    0.161576
4          1539    0.156689
5           964    0.098147
1           938    0.095500
7           777    0.079108
2           731    0.074425
8           730    0.074323
```

The group by result shows that 26% of the zip codes have 24-36% lower level education and 16% of the zip codes have 63-75% lower level education. Our data mostly contains areas with levels 3, 6 and 4 of this attribute.

```
caravan_plot = sns.barplot(x='MKOOPKLA', y='MKOOPKLA', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel="MKOOPLA - Purchasing Power Class", ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the lower level education result. We can see clearly how most of the observations fall into the 3, 6 and 4 of of this attribute.

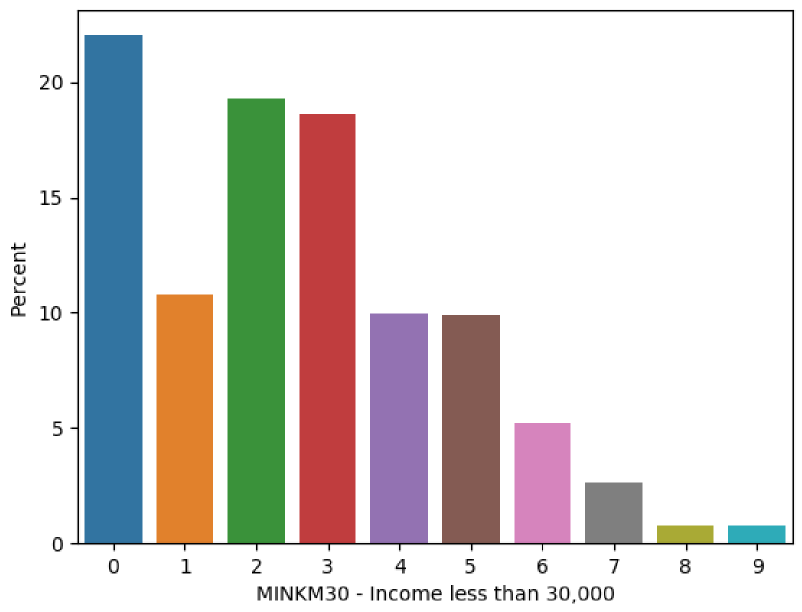
5.5 MINKM30 - Income less than 30,000

```
grouped = red_caravan_df.groupby('MINKM30')
grouped['MINKM30'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[109]:
         count  percentage
MINKM30
0         2164    0.220322
2         1893    0.192731
3         1826    0.185909
1         1060    0.107921
4          981    0.099878
5          971    0.098860
6          512    0.052128
7          261    0.026573
9           78    0.007941
8           76    0.007738
```

The group by result shows that 22% of the zip codes have 0 percentage of income less than 30,000 and 37% have 11-36% of this type of income.

```
caravan_plot = sns.barplot(x='MINKM30', y='MINKM30', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel ="MINKM30 - Income less than 30,000",ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the income less than 30,000 attribute. We can see clearly how most of the observations fall into the 0, 2 and 3 of of this attribute.

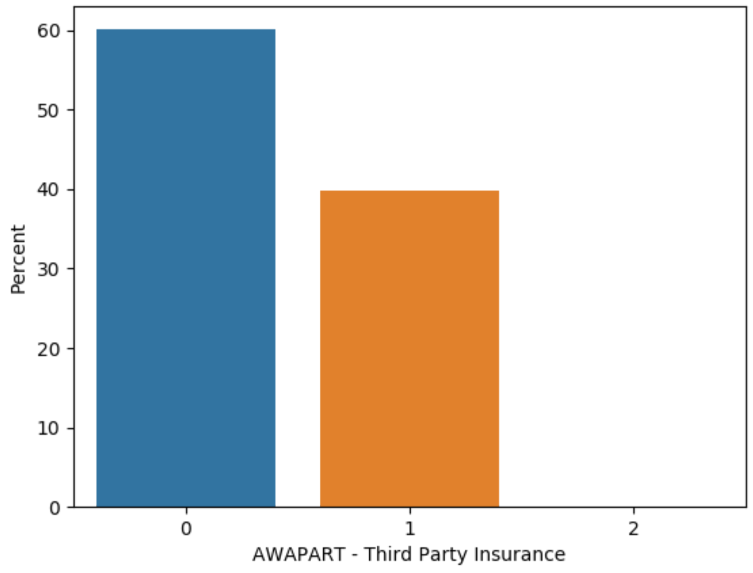
5.6 AWAPART - Third Party Insurance

```
grouped = red_caravan_df.groupby('AWAPART')
grouped['AWAPART'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[111]:
         count  percentage
AWAPART
0         5903    0.600998
1         3909    0.397984
2           10    0.001018
```

The group by result shows that 60% of the zip codes have 0 third party Insurance and 39.8% of the zipcodes have between 1-49 number of third party Insurance.

```
caravan_plot = sns.barplot(x='AWAPART', y='AWAPART', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel ="AWAPART - Third Party Insurance",ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the third party Insurance attribute. We can see clearly how most of the observations most of the observations fall into the 0 and 1 level of this attribute.

5.7 MHHUUR - Rented House

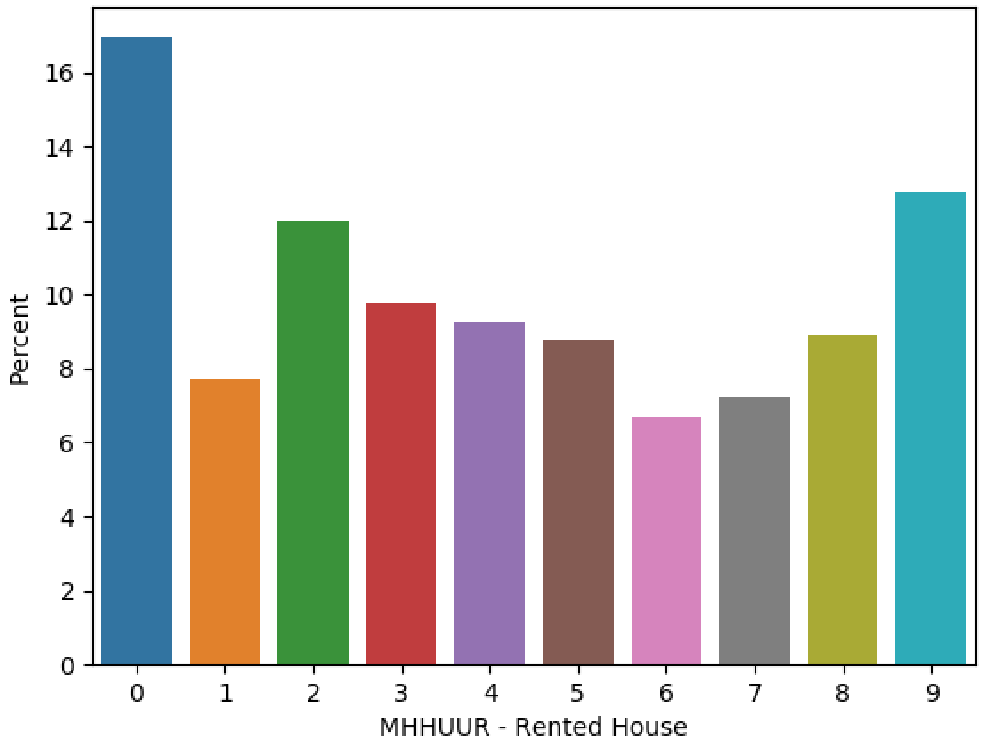
```
grouped = red_caravan_df.groupby('MHHUUR')
grouped['MHHUUR'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[113]:
        count  percentage
MHHUUR
0        1663    0.169314
9        1255    0.127774
2        1177    0.119833
3         961    0.097842
4         908    0.092446
```

```
8          874    0.088984
5          863    0.087864
1          755    0.076868
7          710    0.072287
6          656    0.066789
```

The group by result shows that 17% of the zip codes have 0% of rented house and 12.7% of the zipcodes have 100% number of rented house.

```
caravan_plot = sns.barplot(x='MHHUUR', y='MHHUUR', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel="MHHUUR - Rented House",ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the rented house attribute. We can observe from the graph that 17% of the observations fall into 0 level of this attribute which means that those zipcodes has 0% of rented house and rest of the levels are almost uniformly distributed.

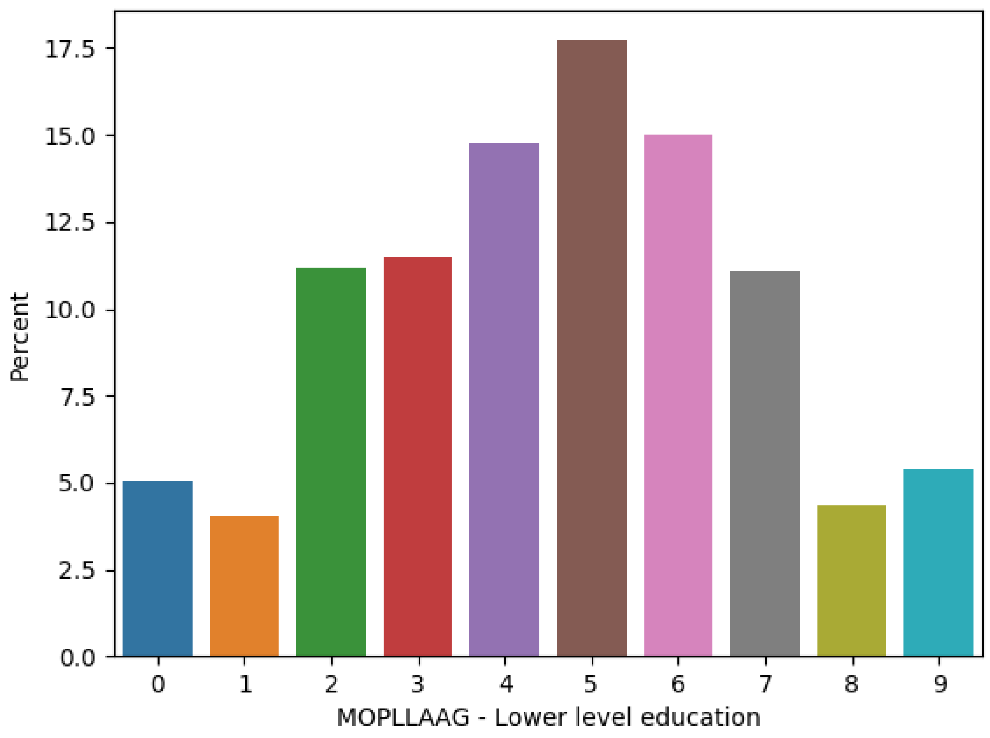
5.8 MOPLLAAG - Lower level education

```
grouped = red_caravan_df.groupby('MOPLLAAG')
grouped['MOPLLAAG'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[115]:
          count  percentage
MOPLLAAG
5          1740    0.177153
6          1472    0.149868
4          1448    0.147424
3          1128    0.114844
2          1099    0.111892
7          1090    0.110975
9           531    0.054062
0           494    0.050295
8           424    0.043168
1           396    0.040318
```

The group by result shows that 17.8% of the zip codes have 50-62% of lower level education people and 15% of the zipcodes have 63-75% number of lower level education people. We see that a high percentage of the people have lower level education in most of the zipcodes.

```
caravan_plot = sns.barplot(x='MOPLLAAG', y='MOPLLAAG', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel="MOPLLAAG - Lower level education",ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the lower level education attribute. We can observe from the graph that most of the levels for this attribute are almost uniformly distributed and most of the zipcodes have people with lower level education.

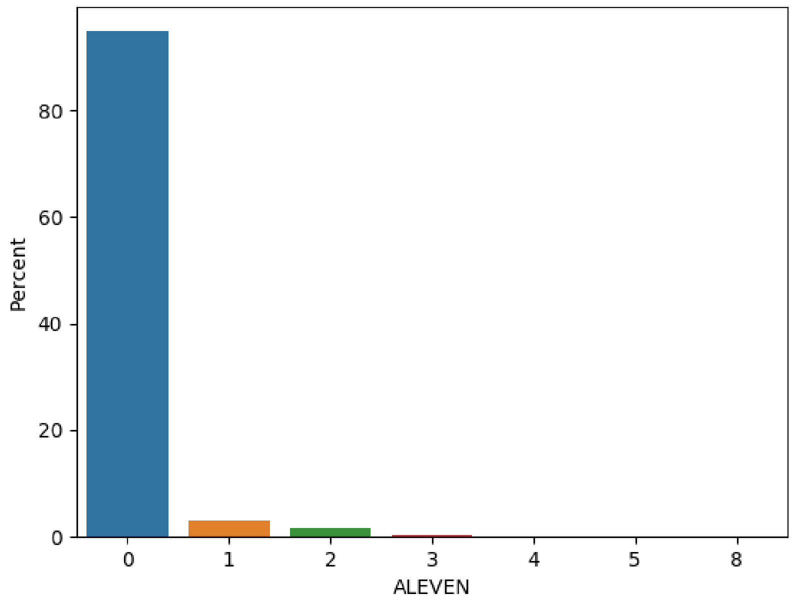
5.9 ALEVEN - Number of life insurances

```
grouped = red_caravan_df.groupby('ALEVEN')
grouped['ALEVEN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[117]:
        count  percentage
ALEVEN
0        9308    0.947668
1         305    0.031053
2         170    0.017308
3          23    0.002342
4          13    0.001324
5           2    0.000204
8           1    0.000102
```

The group by result shows that 95% of the zip codes dont have any Life Insurance policies and about 3.1% of the zipcodes have between 1-49 number of Life Insurance policies .

```
caravan_plot = sns.barplot(x='ALEVEN', y='ALEVEN', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the Life Insurance policy attribute. We can observe from the graph that most of the zipcodes have people with no Life nsurance policy.

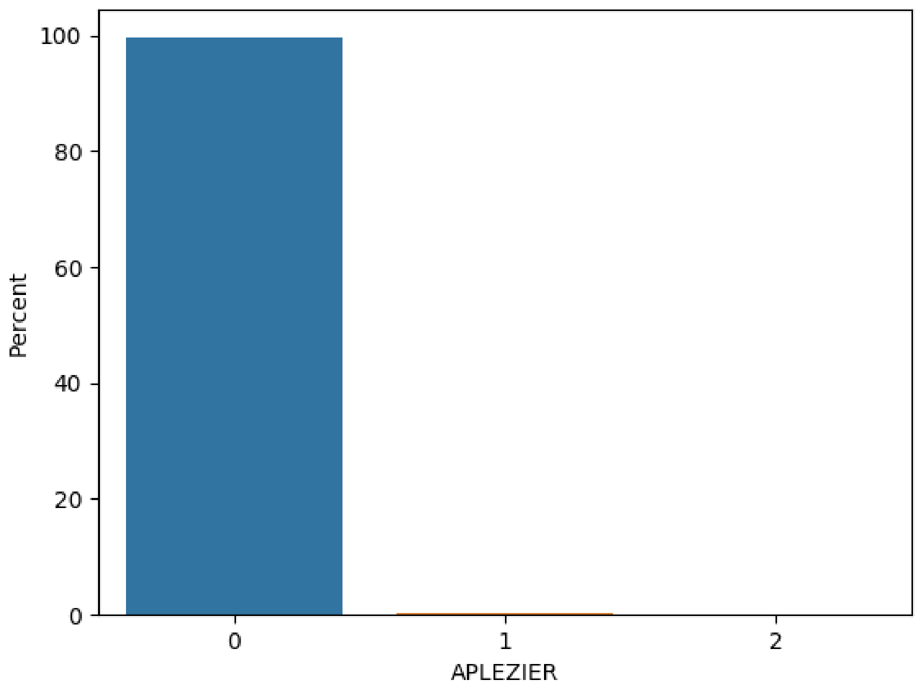
5.10 APLEZIER - Number of boat policies

```
grouped = red_caravan_df.groupby('APLEZIER')
grouped['APLEZIER'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[119]:
          count  percentage
APLEZIER
0          9777    0.995418
1            40    0.004072
2             5    0.000509
```

The group by result shows that 99.5% of the zip codes dont have any Boat Insurance policies and about 0.5% of the zipcodes own between 1-99 number of Boat Insurance policies .

```
caravan_plot = sns.barplot(x='APLEZIER', y='APLEZIER', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the Boat Insurance policy attribute. We can observe from the graph that most of the zipcodes have people with no boat insurance policy.

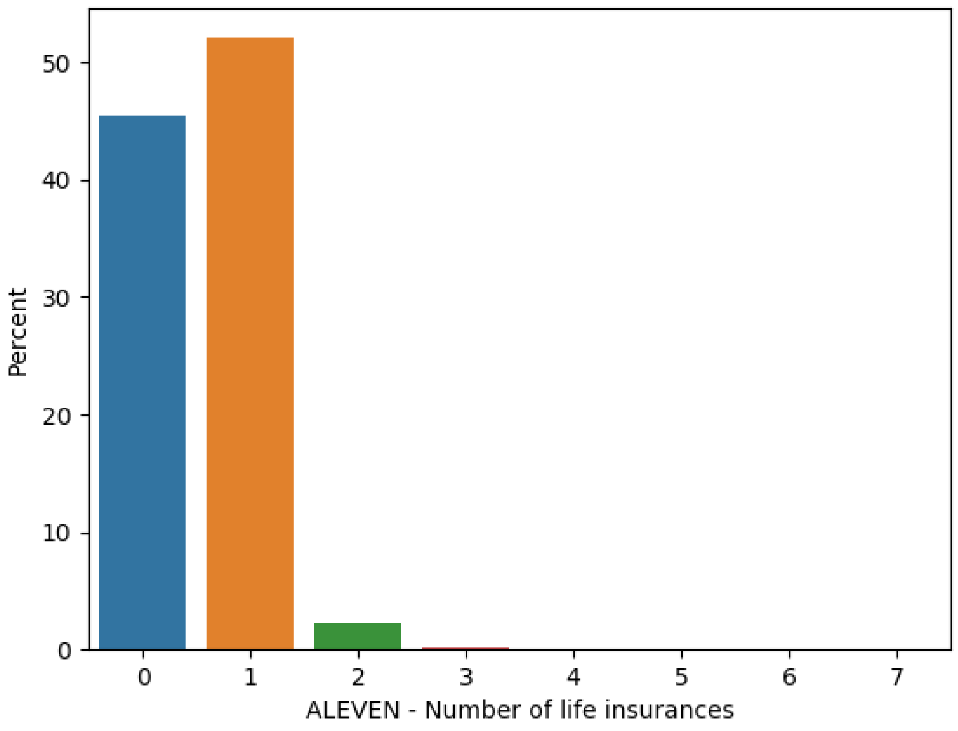
5.11 ABRAND - Number of fire policies

```
grouped = red_caravan_df.groupby('ABRAND')
grouped['ABRAND'].agg({'count':np.size,
                  'percentage': lambda x: len(x) /
len(red_caravan_df)}).sort('percentage', ascending =False)
```

```
Out[121]:
        count  percentage
ABRAND
1        5116    0.520872
0        4464    0.454490
2         221    0.022501
3          11    0.001120
4           6    0.000611
5           2    0.000204
6           1    0.000102
7           1    0.000102
```

The group by result shows that 45.5% of the zip codes dont have any Fire Insurance policies and 53% of zipcodes have between 1-49 any Fire Insurance policies.

```
caravan_plot = sns.barplot(x='ABRAND', y='ABRAND', data=red_caravan_df,
estimator=lambda x: len(x) / len(red_caravan_df) * 100)
caravan_plot.set(xlabel="ALEVEN - Number of life insurances",ylabel="Percent")
display()
plt.gcf().clear()
```



This graph visualizes the percentages of the Fire Insurance policy attribute. We can observe from the graph that almost 45% of the zipcodes have people with no fire insurance policy and rest 54% of the zipcodes have 1-49 number of fire insurance policies.

6. Explore Multiple Variables

The purpose of the following EDA is to give a clear insight into customers having caravan insurance policy and how these customers are different from other customers.

6.1 Caravan vs Demographics

1. Caravan, Average Income and Rented House

```
pd.pivot_table(red_caravan_df[['MHHUUR', 'CARAVAN','MINKGEM']],
               values = 'MINKGEM',
               index  ='MHHUUR',
               columns='CARAVAN',
               aggfunc=np.mean)
```

```
Out[123]:
CARAVAN         0         1
MHHUUR
0        4.540053  4.860606
1        4.313131  4.645161
2        4.320501  4.750000
3        3.979029  4.200000
4        3.861305  4.380000
5        3.557039  3.589744
6        3.342062  3.844444
7        3.206795  3.030303
8        2.966942  3.629630
9        3.057261  3.140000
```

The pivot table below shows the mean of average income for each combination of rented house and caravan insurance policy. The zip codes that bought caravan have average income between 24 to 62%. We can see that of all the zip codes that bought caravan the average income is higher for those that have lower percentage of rented house.

2.Caravan and Purchasing Power Class

```
grouped = red_caravan_df.groupby(['CARAVAN', 'MKOOPKLA'])
grouped['CARAVAN'].agg({'count':np.size,
                    'percentage': lambda x: len(x) / len(red_caravan_df)})
```

```
Out[124]:
                   count  percentage
CARAVAN MKOOPKLA
0       1            906    0.092242
        2            712    0.072490
        3           2435    0.247913
        4           1460    0.148646
        5            909    0.092547
        6           1485    0.151191
        7            674    0.068621
        8            655    0.066687
1       1             32    0.003258
        2             19    0.001934
        3            121    0.012319
        4             79    0.008043
        5             55    0.005600
        6            102    0.010385
        7            103    0.010487
        8             75    0.007636
```

The result above shows all of the combinations between the caravan two levels and the purchasing power class levels in terms of count and percentages. Most zipcodes fall into caravan 0 and purchising power class 3. We learn from these results that 25% of all the zip codes did not purshase caravan and have purchising power class of 24-36%.

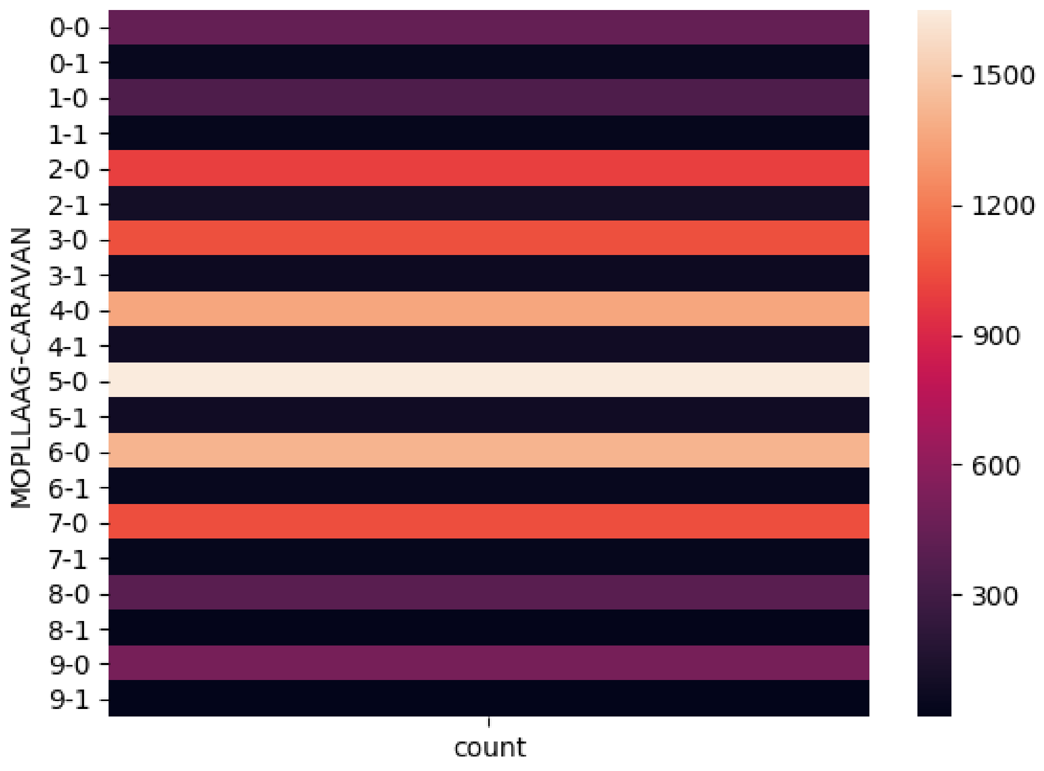
3.Caravan vs Lower Level Education

```
grouped = red_caravan_df.groupby(['CARAVAN', 'MOPLLAAG'])
grouped['CARAVAN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) / len(red_caravan_df)})
```

```
Out[125]:
                  count  percentage
CARAVAN MOPLLAAG
0       0           441    0.044899
        1           356    0.036245
        2           996    0.101405
        3          1055    0.107412
        4          1358    0.138261
        5          1651    0.168092
        6          1416    0.144166
        7          1051    0.107005
        8           399    0.040623
        9           513    0.052230
1       0            53    0.005396
        1            40    0.004072
        2           103    0.010487
        3            73    0.007432
        4            90    0.009163
        5            89    0.009061
        6            56    0.005701
        7            39    0.003971
```

The above table result shows that among the zip codes that purchased caravan policy the first comes the ones with less lower level education.

```
plt.gcf().clear()
grouped= red_caravan_df.groupby(['MOPLLAAG', 'CARAVAN'])
df=pd.DataFrame(grouped['CARAVAN'].agg({'count':np.size}))
sns.heatmap(df)
display()
plt.gcf().clear()
```



The above graph is the visualization of all the combinations of lower level education levels with those who purchased and who did not purchased caravan policy. The lighter the color the larger the number of zipcodes per that combination. The are with the most number of zipcodes correspond to the ones that did not buy caravan insurance and have lower level education of 50-62%. Also the next high number of zipcodes correspond to the ones that have lower level education of 50-75% and did not purchase caravan policy.

6.2 Caravan vs Other Insurances

1. Caravan and Car Insurance Policies

```
grouped = red_caravan_df.groupby(['CARAVAN','APERSAUT'])
grouped['CARAVAN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) / len(red_caravan_df)})
```

```
Out[127]:
                  count  percentage
CARAVAN APERSAUT
0       0          4687    0.477194
        1          4188    0.426390
        2           330    0.033598
        3            19    0.001934
        4             8    0.000814
        5             1    0.000102
        6             1    0.000102
        7             1    0.000102
        12            1    0.000102
1       0           138    0.014050
        1           392    0.039910
        2            54    0.005498
        3             2    0.000204
```

The result above shows all of the combinations between the caravan two levels and the car policies levels in terms of count and percentages. Most zipcodes fall into caravan 0 and car policies levels 0 and 1. We learn from these results that 90% of all the zip codes did not purshase caravan and have car policies between 0-49.

2.Caravan and Life insurances

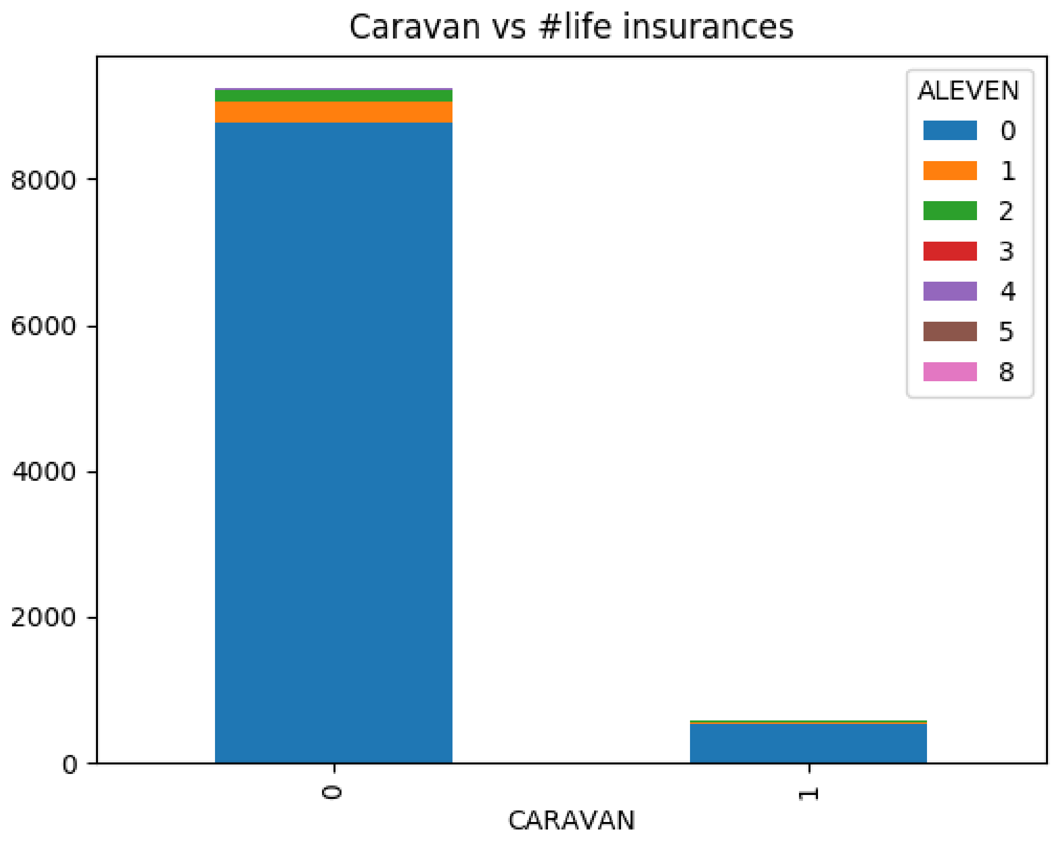
```
grouped = red_caravan_df.groupby(['CARAVAN', 'ALEVEN'])
grouped['CARAVAN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) / len(red_caravan_df)})
```

```
Out[128]:
                count  percentage
CARAVAN ALEVEN
0       0        8772    0.893097
        1         282    0.028711
        2         151    0.015374
        3          19    0.001934
        4           9    0.000916
        5           2    0.000204
        8           1    0.000102
1       0         536    0.054571
```

```
1          23    0.002342
2          19    0.001934
3           4    0.000407
4           4    0.000407
```

The above result shows that out of the zipcodes that fall into caravan = 0, 94% of the zipcodes do not have any life insurance policy. Similar pattern is obersved for zipcodes that fall under caravan = 1.

```
df = red_caravan_df.groupby(['CARAVAN', 'ALEVEN'])
['CARAVAN'].count().unstack('ALEVEN').fillna(0)
df=pd.DataFrame(df)
df.plot(kind='bar',stacked=True,title="Caravan vs #life insurances")
display()
plt.gcf().clear()
```



The above chart shows that irrespective of the zipcode being 0 and 1, most of the zipcodes do not have any life insurance policies

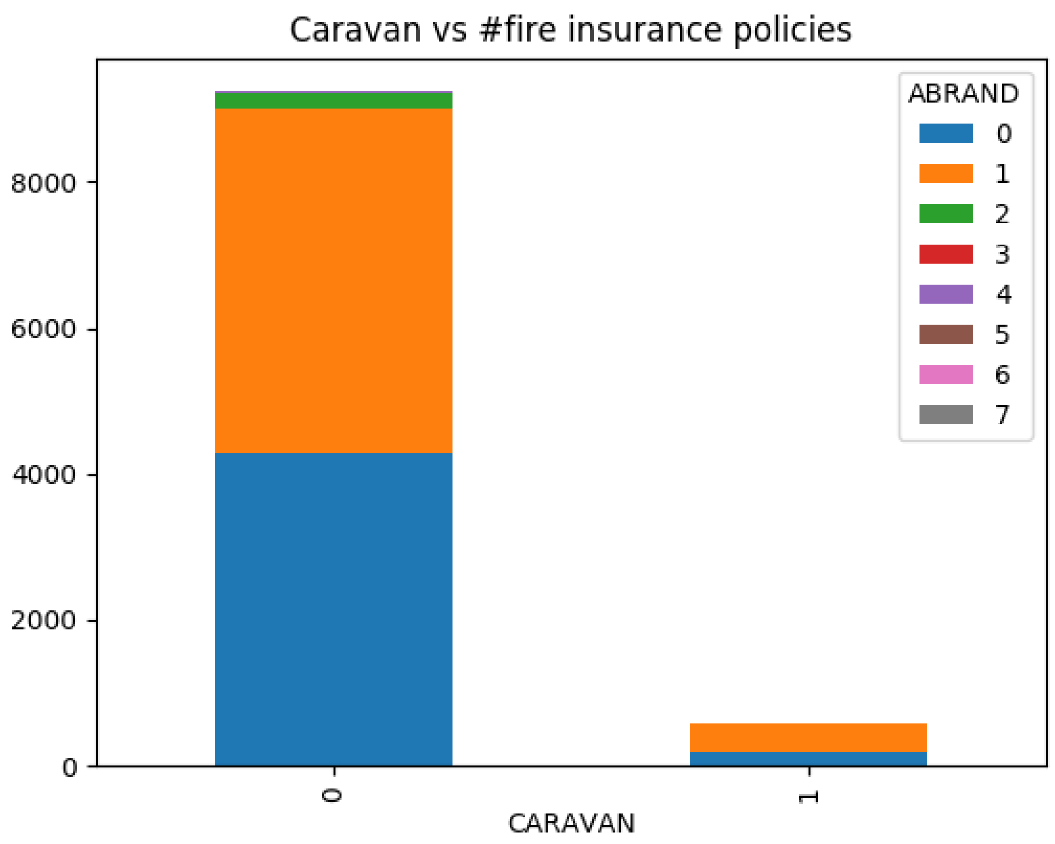
3.Caravan and Fire Insurance Policy

```
grouped = red_caravan_df.groupby(['CARAVAN', 'ABRAND'])
grouped['CARAVAN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) / len(red_caravan_df)})
```

```
Out[130]:
                count  percentage
CARAVAN ABRAND
0       0        4283    0.436062
        1        4724    0.480961
        2         208    0.021177
        3          11    0.001120
        4           6    0.000611
        5           2    0.000204
        6           1    0.000102
        7           1    0.000102
1       0         181    0.018428
        1         392    0.039910
        2          13    0.001324
```

The result above shows all of the combinations between the caravan two levels and the fire policies levels in terms of count and percentages. Most zipcodes that fall into caravan 0 belong car policy levels 0 and 1. We learn from these results that 92% of all the zip codes did not purshase caravan and have fire policies between 0-49.

```
df1 = red_caravan_df.groupby(['CARAVAN', 'ABRAND'])
['CARAVAN'].count().unstack('ABRAND').fillna(0)
my_df1=pd.DataFrame(df1)
my_df1.plot(kind='bar',stacked=True,title="Caravan vs #fire insurance
policies")
display()
plt.gcf().clear()
```



The bove chart shows an interesting analysis compared to the previous chart. Irrespective whether caravan is 0 or 1, we see that there is a good proportion of zipcodes who have fire insurance.

4.Caravan and Third Party Insurance

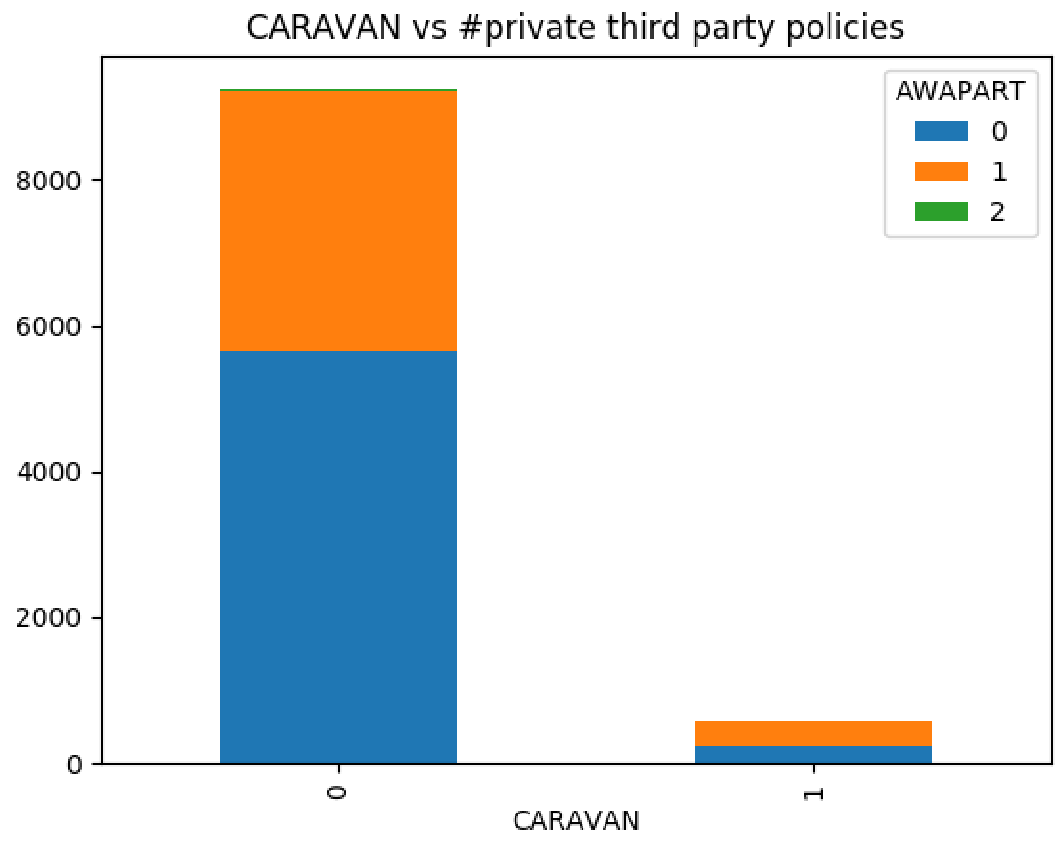
```
grouped = red_caravan_df.groupby(['CARAVAN', 'AWAPART'])
grouped['CARAVAN'].agg({'count':np.size,
                  'percentage': lambda x: len(x) / len(red_caravan_df)})
```

```
Out[132]:
                 count  percentage
CARAVAN AWAPART
0       0         5656    0.575850
        1         3570    0.363470
        2           10    0.001018
```

```
1       0          247     0.025148
        1          339     0.034514
```

The above cross tab shows that 39% of the zipcodes, who fall under carvan = 0 category, have one or more third party insurances. While for zipcodes falling under carvan = 1, almost 58% of the them have one or more third party insurances.

```
df1 = red_caravan_df.groupby(['CARAVAN', 'AWAPART'])
['CARAVAN'].count().unstack('AWAPART').fillna(0)
my_df1=pd.DataFrame(df1)
my_df1.plot(kind='bar',stacked=True,title= "CARAVAN vs #private third party
policies")
display()
plt.gcf().clear()
```



The above chart shows similar trend as fire policies. Almost 39% of the zipcodes, who fall under caravan = 0 category have private third party insurance and 58% for those with carvan = 1.

Conclusion

We started the analysis with the 87 variables that were available initially. We use correlation values at the first step to filter out the significant variables required for the analyses. The selected variables were uncorrelated with each other. Random forests technique was used to identify any other insignificant variables for prediction. This resulted in elimination of other insignificant variables. Some of the important factors using the random forest analyses were #fire policies, #car policies, #life insurances, #boat policies, other third-party insurances, income level and purchasing power.

Then we further drilled down to see how the final shortlisted variables vary with the dependent variable. We used bi-variate analyses including visualizations and frequency reports to see how these independent variables varied with the dependent variable.