2019

Customer's Purchasing Likelihood Prediction

A Machine Learning based project

This project aims at analyzing customer's characteristics and building a machine learned model for predicting likelihood of a customer to purchase a bike and monthly spending with the company.

Prepared by
Arpan Adhikari

RCC Institute of Information Technology
(Department of Computer Science and Engineering)
November, 2019

Abstract

Due to today's transition from visiting physical stores to online shopping, predicting customer behavior in the context of e-commerce is gaining importance. It can increase customer satisfaction and sales, resulting in higher conversion rates and a competitive advantage, by facilitating a more personalized shopping process. By utilizing clickstream and supplementary customer data, models for predicting customer behavior can be built. This study analyzes machine learning models to predict a purchase, which is a relevant use case as applied by a large German clothing retailer. Next, to comparing models this study further gives insight into the performance diferences of the models on sequential clickstream and the static customer data, by conducting a descriptive data analysis and separately training the models on the different datasets. The results indicate that a Random Forest algorithm is best suited for the prediction task, showing the best performance results, reasonable latency, offering comprehensibility and a high robustness. Regarding the different data types, models trained on sequential session data outperformed models trained on the static customer data by far. The best results were obtained when combining both datasets.

Contents

Introduction

1.1 Motivation

Customer analytics is a process by which data from customer behavior is used to help make key business decisions via market segmentation and predictive analytics. This information is used by businesses for direct marketing, site selection, and customer relationship management. Marketing provides services in order to satisfy customers. With that in mind, the productive system is considered from its beginning at the production level, to the end of the cycle at the consumer. Customer analytics plays an important role in the prediction of customer behavior.

1.1.1 Predicting customer behavior

Forecasting buying habits and lifestyle preferences is a process of data mining and analysis. This information consists of many aspects like credit card purchases, magazine subscriptions, loyalty card membership, surveys, and voter registration. Using these categories, consumer profiles can be created for any organization's most profitable customers. When many of these potential customers are aggregated in a single area it indicates a fertile location for the business to situate. Using a drive time analysis, it is also possible to predict how far a given customer will drive to a particular location[citation needed]. Combining these sources of information, a dollar value can be placed on each household within a trade area detailing the likelihood that household will be worth to a company. Through customer analytics, companies can make decisions based on facts and objective data.

1.1.2 Data mining

There are two types of categories of data mining. Predictive models use previous customer interactions to predict future events while segmentation techniques are used to place customers with similar behaviors and attributes into distinct groups. This grouping can help marketers to optimize their campaign management and targeting processes.

1.1.3 Retail Uses

In retail, companies can keep detailed records of every transaction made allowing them to better understand customer behavior in store. Data mining can be practically applied through performing basket analysis, sales forecasting, database marketing, and merchandising planning and allocation. Basket analysis can show what items are commonly bought together. Sales forecasting shows time based patterns that can predict when a customer is most likely to buy a specific kind of item. Database marketing uses customer profile for effective promotions. Merchandising planning and allocation uses data to allow retailers to examine store patterns in locations that are demographically similar to improve planning and allocation as well as create store layouts.

1.2 Problem definition

In 1998, the Adventure Works Cycles company collected a large volume of data about their existing customers, including demographic features and information about purchases they have made. The c

ompany is particularly interested in analyzing customer data to determine any apparent relationships between demographic features known about the customers and the likelihood of a customer purchasing a bike. Additionally, the analysis should endeavor to determine whether a customer's average monthly spend with the company can be predicted from known customer characteristics.

In this project, there are two objectives :
Objective 1: Build a classification model to predict customer purchasing behavior.
Objective 2: Build a regression model to predict customer purchasing behavior.

This data consists of three files, containing data that was collected on January 1st

AdvWorksCusts.csv

Customer demographic data consisting of the following fields:
CustomerID (integer): A unique customer identifier.
Title (string): The customer's formal title (Mr, Mrs, Ms, Miss Dr, etc.)
FirstName (string): The customer's first name.
MiddleName (string): The customer's middle name.
LastName (string): The customer's last name.
Suffix (string): A suffix for the customer name (Jr, Sr, etc.)
AddressLine1 (string): The first line of the customer's home address.
AddressLine2 (string): The second line of the customer's home address.
City (string): The city where the customer lives.
StateProvince (string): The state or province where the customer lives.
CountryRegion (string): The country or region where the customer lives.
PostalCode (string): The postal code for the customer's address.
PhoneNumber (string): The customer's telephone number.
BirthDate (date): The customer's date of birth in the format YYYY-MM-DD.
Education (string): The maximum level of education achieved by the customer:
Partial High School
High School
Partial College
Bachelors
Graduate Degree
Occupation (string): The type of job in which the customer is employed:
Manual
Skilled Manual
Clerical
Management
Professional
Gender (string): The customer's gender (for example, M for male, F for female, etc.)
MaritalStatus (string): Whether the customer is married (M) or single (S).
HomeOwnerFlag (integer): A Boolean flag indicating whether the customer owns their own home (1) or not (0).
NumberCarsOwned (integer): The number of cars owned by the customer.
NumberChildrenAtHome (integer): The number of children the customer has who live at home.

TotalChildren (integer): The total number of children the customer has.
YearlyIncome (decimal): The annual income of the customer.

AW_AveMonthSpend.csv

Sales data for existing customers, consisting of the following fields:
CustomerID (integer): The unique identifier for the customer.
AveMonthSpend (decimal): The amount of money the customer spends with Adventure Works Cycles on average each month.

AW_BikeBuyer.csv

Sales data for existing customers, consisting of the following fields:
CustomerID (integer): The unique identifier for the customer.
BikeBuyer (integer): A Boolean flag indicating whether a customer has previously purchased a bike (1) or not (0).

Literature review

This chapter reviews the results algorithms used by studies concerned with a similar regression and binary classification problem as the one at hand. To create a broader under-standing of the problem and the different algorithms, the algorithms used for solving it are each explained before reviewing their application and performance in literature.

2.1 Regression

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome variable') and one or more independent variables (often called 'predictors', 'covariates', or 'features').

2.1.1 Linear Regression

A linear regression refers to a regression model that is completely made up of linear variables. Beginning with the simple case, Single Variable Linear Regression is a technique used to model the relationship between a single input independent variable (feature variable) and an output dependent variable using a linear model i.e a line.

The more general case is Multi-Variable Linear Regression where a model is created for the relationship between multiple independent input variables (feature variables) and an output dependent variable. The model remains linear in that the output is a linear combination of the input variables. We can model a multi-variable linear regression as the following:

Y = a_1*X_1 + a_2*X_2 + a_3*X_3 ……. a_n*X_n + b

Where a_n are the coefficients, X_n are the variables and b is the bias. As we can see, this function does not include any non-linearities and so is only suited for modeling linearly separable data. It is quite easy to understand as we are simply weighting the importance of each feature variable X_n using the coefficient weights a_n. We determine these weights a_n and the bias b using a Stochastic Gradient Descent (SGD). Check out the illustration below for a more visual picture!

A few key points about Linear Regression:

Fast and easy to model and is particularly useful when the relationship to be modeled is not extremely complex and if you don't have a lot of data.

Very intuitive to understand and interpret.

Linear Regression is very sensitive to outliers.

2.1.2 Polynomial Regression

When we want to create a model that is suitable for handling non-linearly separable data, we will need to use a polynomial regression. In this regression technique, the best fit line is not a straight line. It is rather a curve that fits into the data points. For a polynomial regression, the power of some independent variables is more than 1. For example, we can have something like:

$Y = a_1*X_1 + (a_2)^2*X_2 + (a_3)^4*X_3$ ……. a_n*X_n + b

We can have some variables have exponents, others without, and also select the exact exponent we want for

each variable. However, selecting the exact exponent of each variable naturally requires some knowledge of how the data relates to the output. See the illustration below for a visual comparison of linear vs polynomial regression.

A few key points about Polynomial Regression:
Able to model non-linearly separable data; linear regression can't do this. It is much more flexible in general and can model some fairly complex relationships.
Full control over the modelling of feature variables (which exponent to set).
Requires careful design. Need some knowledge of the data in order to select the best exponents.
Prone to over fitting if exponents are poorly selected.

2.1.3 Ridge Regression

A standard linear or polynomial regression will fail in the case where there is high collinearity among the feature variables. Collinearity is the existence of near-linear relationships among the independent variables. The presence of hight collinearity can be determined in a few different ways:
A regression coefficient is not significant even though, theoretically, that variable should be highly correlated with Y.
When you add or delete an X feature variable, the regression coefficients change dramatically.
Your X feature variables have high pairwise correlations (check the correlation matrix).
We can first look at the optimization function of a standard linear regression to gain some insight as to how ridge regression can help:
min || Xw - y ||²
Where X represents the feature variables, w represents the weights, and y represents the ground truth. Ridge Regression is a remedial measure taken to alleviate collinearity amongst regression predictor variables in a model. Collinearity is a phenomenon in which one feature variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy. Since the feature variables are so correlated in this way, the final regression model is quite restricted and rigid in its approximation i.e it has high variance.
To alleviate this issue, Ridge Regression adds a small squared bias factor to the variables:
min || Xw — y ||² + z|| w ||²
Such a squared bias factor pulls the feature variable coefficients away from this rigidness, introducing a small amount of bias into the model but greatly reducing the variance.
A few key points about Ridge Regression:
The assumptions of this regression is same as least squared regression except normality is not to be assumed.
It shrinks the value of coefficients but doesn't reaches zero, which suggests no feature selection feature

2.1.4 Lasso Regression

Lasso Regression is quite similar to Ridge Regression in that both techniques have the same premise. We are again adding a biasing term to the regression optimization function in order to reduce the effect of collinearity and thus the model variance. However, instead of using a squared bias like ridge regression, lasso instead using an absolute value bias:
min || Xw — y ||² + z|| w ||
There are a few differences between the Ridge and Lasso regressions that essentially draw back to the differences in properties of the L2 and L1 regularization:

Built-in feature selection: is frequently mentioned as a useful property of the L1-norm, which the L2-norm does not. This is actually a result of the L1-norm, which tends to produces sparse coefficients. For example, suppose the model have 100 coefficients but only 10 of them have non-zero coefficients, this is effectively saying that "the other 90 predictors are useless in predicting the target values". L2-norm produces non-sparse coefficients, so does not have this property. Thus one can say that Lasso regression does a form of "parameter selections" since the feature variables that aren't selected will have a total weight of 0.
Sparsity: refers to that only very few entries in a matrix (or vector) is non-zero. L1-norm has the property of producing many coefficients with zero values or very small values with few large coefficients. This is connected to the previous point where Lasso performs a type of feature selection.
Computational efficiency: L1-norm does not have an analytical solution, but L2-norm does. This allows the L2-norm solutions to be calculated computationally efficiently. However, L1-norm solutions does have the sparsity properties which allows it to be used along with sparse algorithms, which makes the calculation more computationally efficient.

2.1.5 ElasticNet Regression

ElasticNet is a hybrid of Lasso and Ridge Regression techniques. It is uses both the L1 and L2 regularization taking on the effects of both techniques:
min || Xw — y ||² + z_1|| w || + z_2|| w ||²
A practical advantage of trading-off between Lasso and Ridge is that, it allows Elastic-Net to inherit some of Ridge's stability under rotation.
A few key points about ElasticNet Regression:
It encourages group effect in the case of highly correlated variables, rather than zeroing some of them out like Lasso.
There are no limitations on the number of selected variables.

2.2 Classification

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.
Here we have the types of classification algorithms in Machine Learning:

2.2.1 Naive Bayes Classifier

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

2.2.2 Nearest Neighbor

The k-nearest-neighbors algorithm is a classification algorithm, and it is supervised: it takes a bunch of labelled

points and uses them to learn how to label other points. To label a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors), and has those neighbors vote, so whichever label the most of the neighbors have is the label for the new point (the “k” is the number of neighbors it checks).

2.2.3 Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. This is better than other binary classification like nearest neighbor since it also explains quantitatively the factors that lead to classification.

2.2.4 Decision Trees

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data.

2.2.5 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

2.2.6 Neural Network

A neural network consists of units (neurons), arranged in layers, which convert an input vector into some output. Each unit takes an input, applies a (often nonlinear) function to it and then passes the output on to the next layer. Generally the networks are defined to be feed-forward: a unit feeds its output to all the units on the next layer, but there is no feedback to the previous layer. Weightings are applied to the signals passing from one unit to another, and it is these weightings which are tuned in the training phase to adapt a neural network to the particular problem at hand.

Machine Learning Process

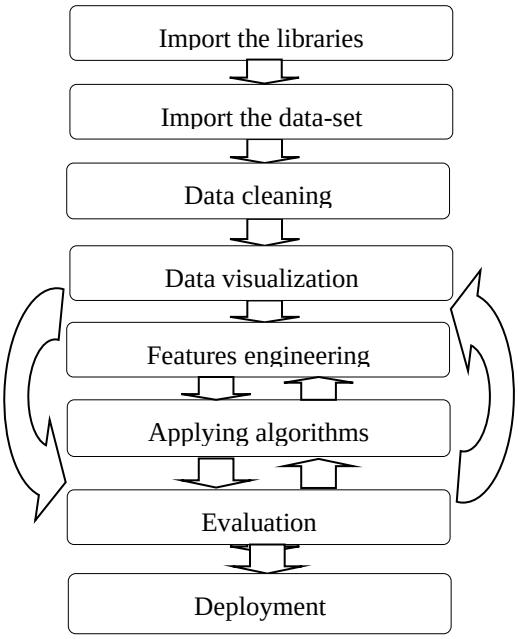

Figure 3.1

3.1 Import the Libraries

```
In [1]:
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5
6  from sklearn.linear_model import LinearRegression,LogisticRegression
7  from sklearn.preprocessing import PolynomialFeatures
8  from sklearn.pipeline import make_pipeline
9  from sklearn.model_selection import train_test_split,KFold,GridSearchCV,cross_val_score
10 from sklearn.metrics import accuracy_score,confusion_matrix,r2_score,mean_squared_error,precision_recall_curve,f1_s
11 from sklearn.externals import joblib
```

Figure 3.2

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

1. A powerful N-dimensional array object
2. Sophisticated (broadcasting) functions
3. Tools for integrating C/C++ and FORTRAN code
4. Useful linear algebra, Fourier transform, and random number capabilities

Pandas is for data manipulation and analysis. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Sklearn is a Python module integrating classical machine learning algorithms in the tightly-knit world of scientific Python packages (numpy, scipy, matplotlib).
It aims to provide simple and efficient solutions to learning problems that are accessible to everybody and reusable in various contexts: machine-learning as a versatile tool for science and engineering.

3.2 Import the Dataset

The training data-set is imported and first five rows and all columns are displayed

```
In [3]: train_data=pd.read_csv('AdvWorksCusts.csv')
        print(train_data.head())
```

```
   CustomerID Title  FirstName MiddleName LastName Suffix  \
0       11000   NaN        Jon          V     Yang    NaN
1       11001   NaN     Eugene          L    Huang    NaN
2       11002   NaN      Ruben        NaN   Torres    NaN
3       11003   NaN    Christy        NaN      Zhu    NaN
4       11004   NaN  Elizabeth        NaN  Johnson    NaN

          AddressLine1 AddressLine2         City StateProvinceName  ...  \
0      3761 N. 14th St          NaN  Rockhampton        Queensland  ...
1           2243 W St.          NaN      Seaford          Victoria  ...
2    5844 Linden Land          NaN       Hobart          Tasmania  ...
3    1825 Village Pl.          NaN   North Ryde   New South Wales  ...
4  7553 Harness Circle          NaN   Wollongong   New South Wales  ...

    BirthDate   Education    Occupation Gender MaritalStatus HomeOwnerFlag  \
0  1966-04-08  Bachelors   Professional      M             M             1
1  1965-05-14  Bachelors   Professional      M             S             0
2  1965-08-12  Bachelors   Professional      M             M             1
3  1968-02-15  Bachelors   Professional      F             S             0
4  1968-08-08  Bachelors   Professional      F             S             1

  NumberCarsOwned NumberChildrenAtHome  TotalChildren  YearlyIncome
0               0                    0              2        137947
1               1                    3              3        101141
2               1                    3              3         91945
3               1                    0              0         86688
4               4                    5              5         92771

[5 rows x 23 columns]
```

Figure 3.3

Dataset Info

```
In [4]: train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16519 entries, 0 to 16518
Data columns (total 23 columns):
CustomerID              16519 non-null int64
Title                   88 non-null object
FirstName               16519 non-null object
MiddleName              9534 non-null object
LastName                16519 non-null object
Suffix                  2 non-null object
AddressLine1            16519 non-null object
AddressLine2            276 non-null object
City                    16519 non-null object
StateProvinceName       16519 non-null object
CountryRegionName       16519 non-null object
PostalCode              16519 non-null object
PhoneNumber             16519 non-null object
BirthDate               16519 non-null object
Education               16519 non-null object
Occupation              16519 non-null object
Gender                  16519 non-null object
MaritalStatus           16519 non-null object
HomeOwnerFlag           16519 non-null int64
NumberCarsOwned         16519 non-null int64
NumberChildrenAtHome    16519 non-null int64
TotalChildren           16519 non-null int64
YearlyIncome            16519 non-null int64
dtypes: int64(6), object(17)
memory usage: 2.9+ MB
```

Figure 3.3

Labels

Labels are the data which are need to be predicted as these are classification and regression model training labels are provided so that a model can be built with the help of labeled data.

```
In [7]:
label_data1=pd.read_csv('AW_AveMonthSpend.csv')
label1=label_data1['AveMonthSpend']
print(label1.head())

0     89
1    117
2    123
3     50
4     95
Name: AveMonthSpend, dtype: int64
```

Figure 3.4

```
In [8]:
label_data2=pd.read_csv('AW_BikeBuyer.csv')
label2=label_data2['BikeBuyer']
print(label2.head())

0    0
1    1
2    0
3    0
4    1
Name: BikeBuyer, dtype: int64
```

Figure 3.5

3.3 Data cleaning

Data cleaning is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data cleaning is a proven method of resolving such issues.

In Real world data are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

3.3.1 Handling Duplicate Values

Duplicate values must be removed properly before implementing algorithms as it can ruin training process. Identifying duplicate values is a little difficult task as multiple data in some feature may have same values so one way to correctly identify duplicity is by using a feature which is known to have unique value like in this case it is 'CustomerID'. So finding duplicate rows by comparing 'CustomerID' and removing the whole row is a good choice as no two Customer ID can be same. See below the figure for better understanding.

```
In [45]:
train_data.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
label_data1.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
label_data2.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
print(train_data.duplicated().sum())

0
```

Figure 3.6

3.3.2 Handling Missing Values

From figure 3.2 it can be observe that there are total of 16519 rows and 23 columns (6 integer and 17 object), there are some missing values in column 'Title','middle name','suffix' and 'Address line 2'. Since there are few non null values in these 4 columns as compared to other columns these columns can be removed from dataset as they are not providing much information and also there are enough data in

dataset.

3.3.2 Removing useless columns

There are some useless columns in the data set are 'CustomerID', 'Title', 'FirstName', 'MiddleName', 'LastName', 'City', 'StateProvinceName', 'Suffix', 'AddressLine2', 'AddressLine1', 'PostalCode', 'PhoneNumber', 'BirthDate'. Among these 'BirthDate' is used for determining 'age' which proved to be an important feature and 'CustomerID' is used to treat duplicate values but after that there's no need to keep them or they will corrupt training algorithm as they are uninterruptable features and converting them to numerical values is costly for memory. See below the output after removing these features.

```
In [5]:
train_data.drop(columns=['CustomerID','Title','FirstName','MiddleName','LastName','City','StateProvinceName',
                         'Suffix','AddressLine2','AddressLine1','PostalCode','PhoneNumber','BirthDate'],inplace=True)
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16519 entries, 0 to 16518
Data columns (total 10 columns):
CountryRegionName       16519 non-null object
Education               16519 non-null object
Occupation              16519 non-null object
Gender                  16519 non-null object
MaritalStatus           16519 non-null object
HomeOwnerFlag           16519 non-null int64
NumberCarsOwned         16519 non-null int64
NumberChildrenAtHome    16519 non-null int64
TotalChildren           16519 non-null int64
YearlyIncome            16519 non-null int64
dtypes: int64(5), object(5)
memory usage: 1.3+ MB
```

Figure 3.7

3.4 Data Visualization

After getting an overview of dataset our next task is data visualization to study data in more details.

```
In [ ]:
def vizual(data,columns,label=label2,type='bar'):
    if type=='bar':
        for col in columns:
            sns.countplot(y=data[col])
            plt.show()
    if type=='hist':
        for col in columns:
            sns.distplot(data[col],bins=30)
            plt.show()
    if type=='scatter':
        for col in columns:
            sns.scatterplot(col,label,data=data,s=10)
            plt.show()
    if type=='violin':
        for col in columns:
            if(data[col].dtype=='object'):
                sns.violinplot(col,label,data=data)
                plt.show()
            else:
                sns.violinplot(label,col,data=data)
                plt.show()

print('\n\t\tData Vizualization\n\n\tBar Plot')
vizual(train_data,cat_col,type='bar',label=label2[label2==1])

print('\tHistogram Plot')
vizual(train_data,num_col,type='hist',label=label2[label2==1])
print('\tScatter Plot')
vizual(train_data,num_col,type='scatter',label=label1)
print('\tViolin Plot')
vizual(train_data,train_data.columns,type='violin')
```

3.4.1 Bar plots

Bar plots are excellent visualization tools for observing distribution of buyers among multiple groups.

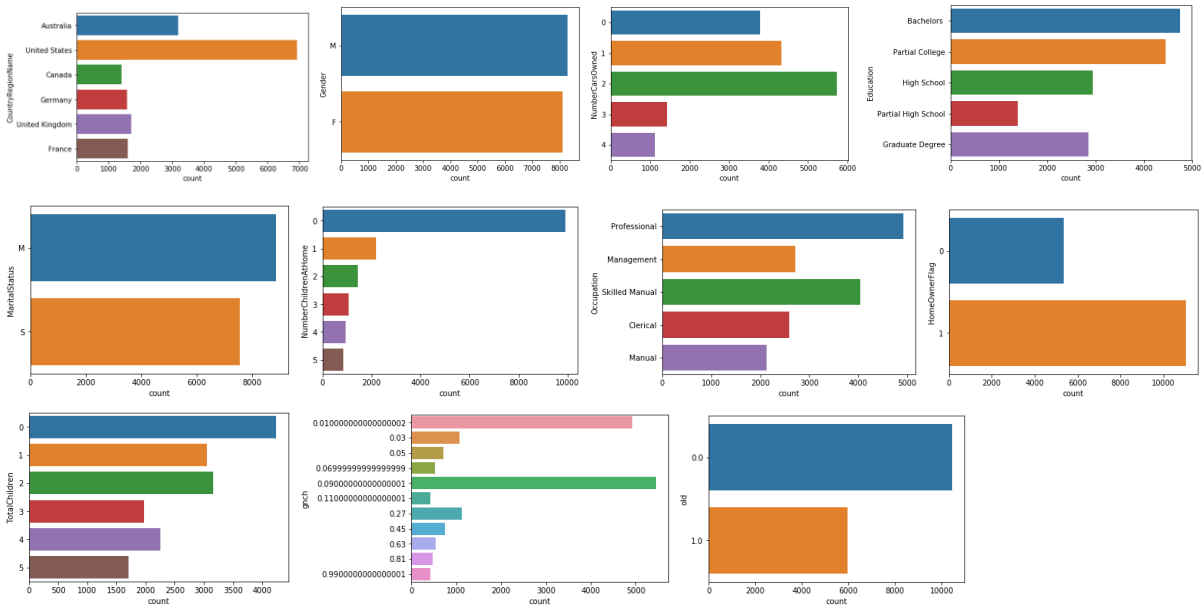

Figure 3.1

Some hypothesis can be generated by analyzing bar plots in Figure 3.1. It is observe that among all the countries United States are the largest buyers followed by Australia and Canada are smallest buyer, proffesionals are largest buyers followed by skilled manuals and manuals are smallest buyers. Similarly more hypothesis are generated by observing all the bar plots in Figure 3.1.

3.4.2 Density Plots

Density plots are used to observe density of a feature (divided into certain intervals) among the intervels.

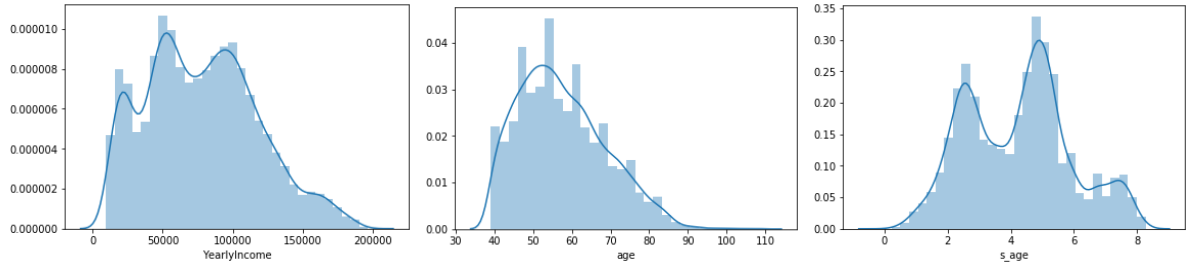

Figure 3.2

Figure 3.2 shows that most of the bike buyers belongs to the age group 40 to 60 above that there are very few buyers. It can also be seen that yearly income of buyer can be divided into four groups low income (around 10000), middle income (around 50000), high income (around 100000) and very high income (above 100000). Some new features can be constructed by divide age and yearly income into groups.

3.4.3 Scatter Plots

Scatter plots is used to measure the variation of one variable(feature) with respect to another variable(label).

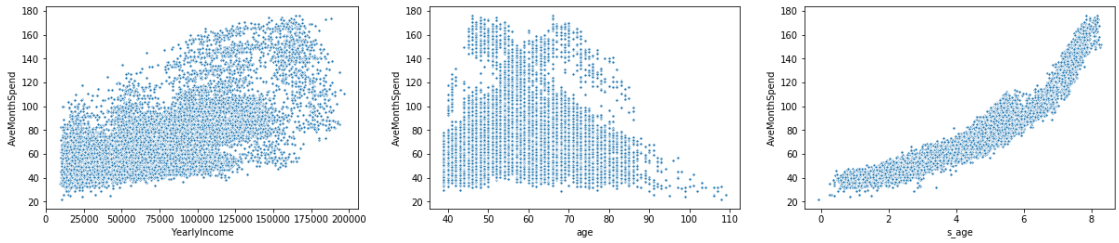

Figure 3.3

It is observe from the scatter plots that average monthly spending of customers' increases as yearly income increase and age decreases, though they are not so strictly depend but the two features can be combined to get more dependency of labels on features. Average monthly spending seems to be more depended on 's_age' which is combination of age and yearly income.

3.4.4 Violin Plots

A violin plot plays a similar role as a box and whisker plot. It shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared. Unlike a box plot, in which all of the plot components correspond to actual datapoints, the violin plot features a kernel density estimation of the underlying distribution.

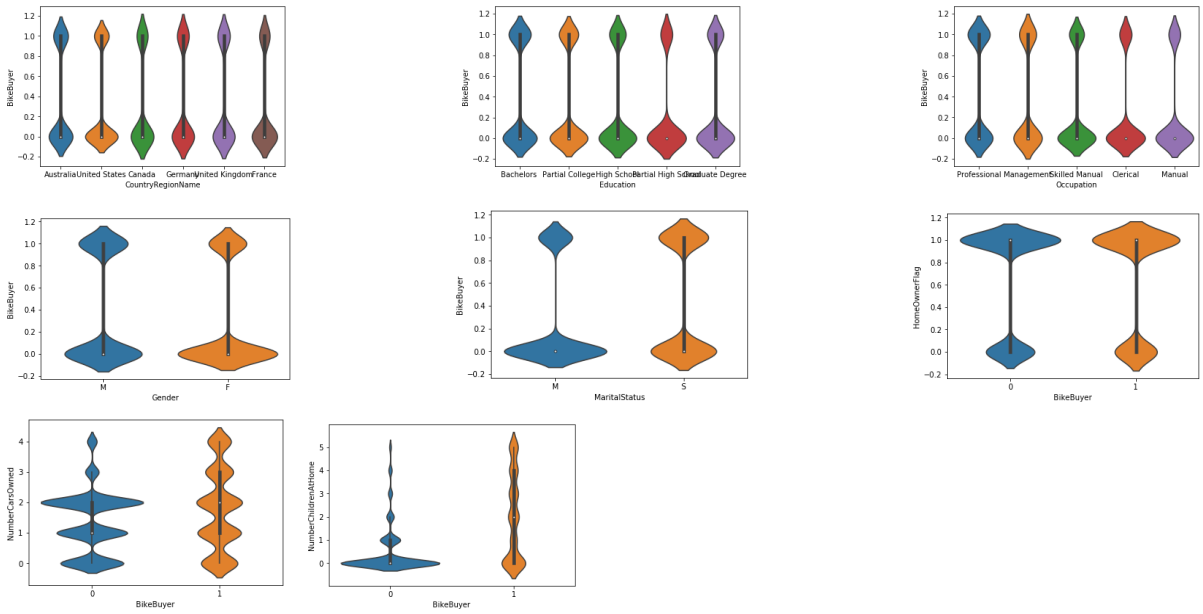

Figure 3.4

3.5 Features Engineering

Feature engineering is the technique of modifying features and making them more useful for prediction.

```
In [ ]:
def features_engg(data):
    data['Gender'].replace({'M':0.9,'F':0.1},inplace=True)
    data['NumberChildrenAtHome'].replace({0:0.1,1:0.3,2:0.5,3:0.7,4:0.9,5:1.1},inplace=True)
    data['s_age']=data['Gender']*data['YearlyIncome']/data['age']*data['NumberChildrenAtHome']
    data['gnch']=data['Gender']*data['NumberChildrenAtHome']

    data['old']=data['age'].apply(lambda x:0 if x<=60 else 1)

    data['NumberChildrenAtHome'].replace({1:0,2:1,3:1,4:1,5:1},inplace=True)
    data['NumberCarsOwned'].replace({1:0,2:0,3:1,4:1},inplace=True)
    data['TotalChildren'].replace({1:0,2:0,3:1,4:1,5:1},inplace=True)

    data=pd.get_dummies(data)

    data=scale(data,columns=['s_age'],type='log')
    data=scale(data,data.select_dtypes('int64','float64'))

    return data

train_data=features_engg(train_data)

print('\n\t\tData Vizualization After Data Modification\n\n\tHistogram Plot')
vizual(train_data,['s_age'],type='hist')
print('\tScatter Plot')
vizual(train_data,['s_age'],type='scatter',label=label1)
print('\tBar Plot')
vizual(train_data,['gnch','old'],type='bar')
print('\tViolin Plot')
vizual(train_data,['gnch','old'],type='violin')
```

Figure 3.5

3.5.1 Features Generation

Feature Generation is a technique of generating new features with existing feature. Here three new features are generated namely ‘age’,‘s_age’, ’old’. The formulation of these features are given below

[‘age’]=2019-[‘year of birth’]

[‘s_age’]= ['Gender']*['YearlyIncome']/['age']*['NumberChildrenAtHome']

['gnch']=['Gender']*['NumberChildrenAtHome']

3.5.2 Features Discretization

Features Dicritization is a technique of replacing the raw values of numeric attribute by interval levels or conceptual levels. A new feature called ‘old’ is generated by discretizing features ‘age’

[‘old’] :

[‘age’<60] → 0

[‘age’>60] → 1

3.5.3 Features Binning

Features Binning is a technique of grouping the raw values categorical variables into separate groups(bins) according to their closest bin value.

['NumberCarsOwned'] :

1,2 → 0

3,4 → 1

['TotalChildren'] :

1,2 → 0

3,4,5 → 1

['NumberChildrenAtHome'] :

1 → 0

2,3,4,5 →1

3.5.4 Transform Categorical Values

Since, machine learning models are based on Mathematical equations and it can be intuitively understand that it would cause some problem if we can keep the Categorical data in the equations because we would only want numbers in the equations. So, we need to encode the Categorical Variable. Here we have use dummy variables to achieve encoding. Dummy Variables is one that takes the value 0 or 1 to indicate the absence or presence of some categorical effect that may be expected to shift the outcome. Instead of having one column per feature, we are going to have multiple columns per feature.

Number of Columns = Number of Categories.

3.5.5 Features Scaling

Feature scaling is the method to limit the range of variables so that they can be compared on common grounds. In this dataset the 's_age' are scaled using 'logarithmic' scaling and rest are scaled using 'min-max' scaling.

3.6 Applying algorithms

In any Machine Learning model before applying algorithms the dataset must be split into two separate sets

1. Training Set
2. Test Set

We split the data-set into 80:20 what does it mean, 80 percent data take in train and 20 percent data take in test. However, this Splitting can be varies according to the data-set shape and size.

X_train is the training part of the matrix of features.

X_test is the test part of the matrix of features.

y_train is the training part of the dependent variable that is associated to X_train here.

y_test is the test part of the dependent variable that is associated to X_train here.

3.6.1 Regression model

Regression model is used to predict numerical labels. Some of the commonly used regression models are Linear regression, Lasso regression, Ridge regression, Elastic regression etc. Among them Linear regression has performed best both in term of accuracy and root mean squared error.

```
In [ ]:
def regression(x,y=label1,test_size=0.2):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=test_size)
    model=make_pipeline(PolynomialFeatures(degree=2),LinearRegression())
    model.fit(x_train,y_train)
    return model,x_test,y_test
```

Figure 3.6

3.6.2 Classification model

Classification model are used to predict class labels. Some of the commonly used classification models are Logistic regression, Decision tree, Random forest, Adaboost classifier, Neural network etc. Among them Logistic regression model perform best in all prospect (accuracy score, F1 score, ROC AUC score, PR AUC score).

```
In [ ]:
def classify(x,y=label2,test_size=0.2):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=test_size)
    model=make_pipeline(PolynomialFeatures(degree=2),LogisticRegression(C=0.7,penalty='l1',class_weight='balanced',
                                                                        solver='liblinear'))
    model.fit(x_train,y_train)
    return model,x_test,y_test
```

Figure 3.7

3.7 Evaluation

Evaluation is the final step for any machine learning project before deploying the model for use. Model evaluation aims to estimate generalized accuracy of a model on future unseen data. Method for evaluating model performance are divided into 2 categories namely holdouts and cross-validation.
The purpose of holdouts is to test a model on different data (test set) than it was trained on.
The most common cross-validation technique is k-fold cross-validation where the original dataset is partitioned into k equal size subsamples, called folds. This is repeated k times, such that every times one of the k subset is used as the test set/validation set and other k-1 subsets are put together to form a training set.

3.7.1 Regression

The following output shows actual vs predicted values and it can be seen there are very small difference between actual and predicted values.

```
In [25]:  1 reg_model,x_test,y_test=regression(train_data)
          2 print(predict(reg_model,x_test,y_test).head(10))
```

	Actua Value	Predicted Value
3356	76	71.902832
15563	49	48.801453
14595	87	88.901764
4544	67	65.311493
4955	46	46.188354
12656	68	67.735199
11480	54	54.195465
5825	54	48.134399
4135	45	44.506989
4697	51	51.809570

Figure 3.8

r2_score

r2_score (y_true, y_pred, sample_weight=None,multioutput="uniform_average")
Type: Function in sklearn.metrics.regression module
R^2 (coefficient of determination) regression score function.
Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.
R2 score = 0.9869264404436823

Root mean squared error

Root mean squared error is the calculated by the formula $\sqrt{(\sum(\text{actual value-predicted value})^2)}$
Root Mean Squared Error= 3.165729104616804

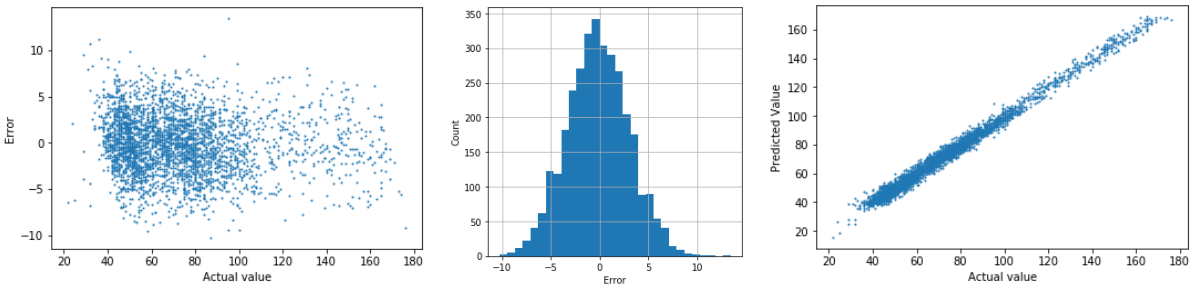

Figure 3.9

Mean R2 score after cross validation = 0.9870290692054745

3.7.2 Classification

The following output shows actual vs predicted values and it can be seen there are only two misclassified among actual and predicted values.

```
In [33]:  1 class_model,x_test,y_test=classify(train_data,label2)
          2 print(predict(class_model,x_test,y_test).head(10))
```

	Actua Value	Predicted Value
10048	1	1
13767	1	1
12238	0	0
10234	0	1
5588	1	1
10877	1	1
7879	0	1
5065	1	1
7225	0	0
8299	1	1

Figure 3.10

Accuracy score is calculated by the formula ∑(correctly predicted value/total values)
Accuracy score= 0.7744590064004877

The **F1 score** can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:
F1 = 2 * (precision * recall) / (precision + recall)
In the multi-class and multi-label case, this is the average of the F1 score of each class with weighting depending on the average parameter.

The **precision** is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The **recall** is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

ROC AUC computes Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
Note: this implementation is restricted to the binary classification task or multilevel classification task in label indicator format.

	F1 Score	PR AUC	ROC AUC
No Skill:	0.0	0.667784212130448	0.5
Logistic:	0.7011308562197093	0.7927325151583599	0.8660912931530136

Confusion matrix evaluate the accuracy of a classification
By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in I group but predicted to be in group j.
Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and

false positives is $C_{0,1}$.

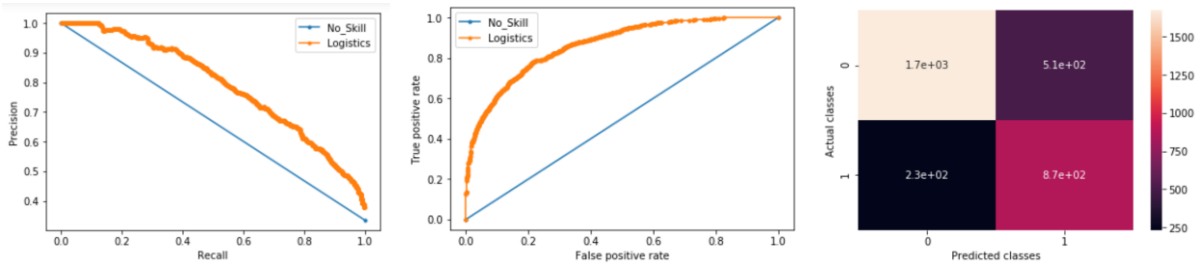

Figure 3.11

Mean ROC AUC after cross validation = 0.8587612203915918

3.8 Deployment

In machine learning it is not a good idea to train the model again and again at each time whenever a new unknown test-set is encounter which is memory as well as CPU wastage rather it is preferable to save the model after training so that it can be use whenever needed and no need to train again and again. This task of saving model is done by a function 'joblib' from 'sklearn.externels' package

3.8.1 Regression model

```
In [61]:  1  joblib.dump(reg_model,'reg_model.sav')
Out[61]: ['reg_model.sav']
```

Figure 3.12

3.8.2 Classification model

```
In [62]:  1  joblib.dump(class_model,'class_model.sav')
Out[62]: ['class_model.sav']
```

Figure 3.13

Application

The machine learning models are ready now it's time to predict some value on unknown dataset using saved models. This process is much similar to training process except we can skip some of training step. A flow chart of the process is given below.

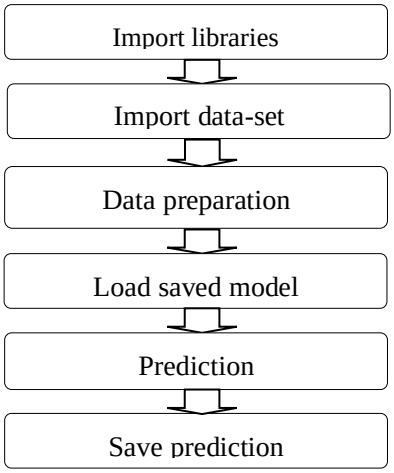

Figure 4.1

4.1 Import libraries

This is same as the training process we need to import some libraries but this time much fewer in number as compare to training process.

```
In [ ]:
1 import pandas as pd
2 import numpy as np
3 from sklearn.externals import joblib
```

Figure 4.2

4.2 Import data-set

The unknown data-set (test-set) on which prediction should be done using newly built models are imported and the data-set looks similar to training data-set.

```
In [ ]:
1 test_data=pd.read_csv('AW_test.csv')
2 test_data.info()
```

Figure 4.3

4.3 Data preparation

In this step test-data is been modified as same as train-data. At first data cleaning is done followed by features engineering.

4.3.1 Data cleaning

```
In [ ]:
test_data.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
features=test_data.copy()
features['age']=(features['BirthDate'].apply(lambda x:x[-4:])).apply(lambda x:2019-np.int64(x))
features.drop(columns=['CustomerID','Title','FirstName','MiddleName','LastName','City','StateProvinceName','Suffix'
                       ,'AddressLine2','AddressLine1','PostalCode','PhoneNumber','BirthDate'],inplace=True)
```

Figure 4.4

4.3.2 Features engineering

```
In [ ]:
def preperation(data):
    data['Gender'].replace({'M':0.9,'F':0.1},inplace=True)
    data['NumberChildrenAtHome'].replace({0:0.1,1:0.3,2:0.5,3:0.7,4:0.9,5:1.1},inplace=True)

    data['s_age']=data['Gender']*data['YearlyIncome']/data['age']*data['NumberChildrenAtHome']
    data['old']=data['age'].apply(lambda x:0 if x<=60 else 1)
    data['gnch']=data['Gender']*data['NumberChildrenAtHome']

    data['NumberChildrenAtHome'].replace({1:0,2:1,3:1,4:1,5:1},inplace=True)
    data['NumberCarsOwned'].replace({1:0,2:0,3:1,4:1},inplace=True)
    data['TotalChildren'].replace({1:0,2:0,3:1,4:1,5:1},inplace=True)

    data=pd.get_dummies(data)
    data=scale(data,columns=['s_age'],type='log')
    data=scale(data,data.select_dtypes('int64','float64'))

    return data
features=preperation(features)
```

Figure 4.5

4.4 Load saved models

Previously in training we already saved our models and now we can used that just by loading them thus saving memory, CPU and time from further training.

```
In [ ]:
reg_model=joblib.load('reg_model.sav')
class_model=joblib.load('class_model.sav')
```

Figure 4.6

4.5 Prediction

After performing all sorts of data cleaning and features engineering on test-data our final task is to predict the labels which are ‘Average Monthly Spending’ and ‘Bike Buyer’ for every customer in test-set.

```
In [ ]:
test_data['Avarage Monthly Spending']=reg_model.predict(features)
test_data['Bike Buyer']=class_model.predict(features)
```

Figure 4.6

4.6 Save prediction

After prediction it is important to save the predicted value along with the data for later use. Here prediction are saved in a file ‘Result.csv’

```
In [ ]:
  1  test_data.to_csv('Result.csv')
```

Figure 4.7

4.6.1 Result.csv

The screenshot below shows some rows of ‘Result.csv’ where the last two columns are prediction.

	Custome	Title	FirstNam	MiddleNa	LastNam	Suffix	Address	Address	City	StatePro	Country	PostalCo	PhoneNu	BirthDate	Educatio	Occupati	Gender	MaritalSt	HomeOw	Number(	Number(	TotalChil	YearlyInc	Avarage	Bike Buyer
0	18988		Courtney	A	Baker		8727 Buena Vista A		Fremont	Californi	United St	94536	133-555-0	#######	Bachelor	Manager	F	S	0	2	0	5	86931	39.9694	0
1	29135		Adam	C	Allen		3491 Cook Street		Haney	British Co	Canada	V2W 1W	252-555-0	#######	Bachelor	Skilled M	M	M	1	2	2	4	100125	112.912	1
2	12156		Bonnie		Raji		359 Pleasant Hill R		Burbank	Californi	United St	91502	409-555-0	#######	Graduate	Manager	F	M	1	2	0	4	103985	31.7516	0
3	13749		Julio	C	Alonso		8945 Euclid Ave.		Burlingar	Californi	United St	94010	175-555-0	9/22/1958	Graduate	Skilled M	M	M	1	0	0	4	127161	84.228	0
4	27780		Christy	A	Andersen		42, boulevard Tren		Dunkerq	Nord	France	59140	1 (11) 500	3/19/1965	High Sch	Manual	F	M	1	1	2	2	21876	58.197	0
5	16375		Gabriella		Hernandez		5689 Almondtree C		Spring V	Californi	United St	91977	165-555-0	8/23/1957	Partial Co	Clerical	F	M	1	1	0	1	44467	46.3101	0
6	16109		Adam	K	Turner		9800 American Bea		N. Vanco	British Co	Canada	V7L 4J4	222-555-(	#######	Bachelor	Skilled M	M	S	1	2	2	4	77702	97.7452	1
7	15606		Marco	C	Prasad		8523 Rios C		Cranbou	Victoria	Australia	3977	1 (11) 500	#######	Bachelor	Professic	M	M	1	3	4	4	99418	141.632	1
8	20888		Clayton	P	Raje		4, place du Tertre		Trembla	Seine Sa	France	93290	1 (11) 500	10/21/1966	Partial Hi	Manual	M	S	1	1	4	4	13522	95.1577	1
9	20716		Kristina	V	Arun		Waldstr 29		Hannove	Nordrhei	Germany	30601	1 (11) 500	9/13/1973	Bachelor	Skilled M	F	M	1	1	1	1	46264	55.6442	0
10	21311		Jennifer	E	Ramirez		7556 Garcia Ranch		Royal Oa	British Co	Canada	V8X	626-555-(	10/27/1955	High Sch	Professic	F	S	1	1	1	2	96375	54.8079	0
11	21741		Alexa	E	Sanders		1133 Fillet Ave		N. Vanco	British Co	Canada	V7L 4J4	263-555-(	6/14/1956	Partial Co	Professic	F	M	1	2	0	3	77688	50.7164	0
12	20101		Devin		Hayes		5171 Polk St.		Oregon C	Oregon	United St	97045	644-555-(	12/6/1956	High Sch	Skilled M	M	M	1	2	0	2	52127	69.122	0
13	28302		Caroline	C	Hughes		4561 Thornhill Plac		Metchosi	British Co	Canada	V9	366-555-(	2/20/1949	High Sch	Professic	F	M	1	2	0	4	60978	45.1665	0
14	14515		Anna		Cooper		6968 Wren Ave.		Lemon G	Californi	United St	91945	115-555-0	#######	Partial Co	Skilled M	F	S	1	1	0	0	36915	40.443	0
15	11674		Morgan	B	Rogers		1168 Escobar		Woodlan	Californi	United St	91364	405-555-(	8/18/1966	Graduate	Professic	F	M	1	0	0	0	86445	54.7882	0
16	11539		Justin		Washington		6092 Chestnut		Concord	Californi	United St	94519	701-555-0	8/22/1959	Graduate	Skilled M	M	M	1	0	0	4	113658	83.5627	0
17	14290		Jonathan		Wilson		8582 Condor Dr.		Corvallis	Oregon	United St	97330	862-555-(	#######	Partial Hi	Skilled M	M	S	1	2	0	2	83326	70.8015	0
18	20615		Laura		Wu		4746 Cle # 61		Matravill	New Sou	Australia	2036	1 (11) 500	10/27/1973	Partial Co	Manager	F	S	1	3	5	5	142823	99.0469	1
19	14818		Diane	F	Ruiz		12, rue Descartes		Colombe	Hauts de	France	92700	1 (11) 500	5/26/1964	Partial Co	Clerical	F	S	1	2	2	2	42704	58.7639	0
20	26973		Alberto	W	Rowe		9650 Valley View F		Cheltenh	England	United Ki	GL50	1 (11) 500	5/27/1973	Partial Hi	Manual	M	M	1	2	0	0	31323	66.6222	0
21	17825		Clayton	F	Li		Heideweg 2459		Ingolstac	Bayern	Germany	85049	1 (11) 500	10/15/1963	Bachelor	Skilled M	M	M	1	0	0	1	61350	75.1975	0
22	15133		Raul	R	Rai		6384 Euclid Ave.		Brisbane	Queensla	Australia	4000	1 (11) 500	#######	Partial Hi	Professic	M	M	1	4	5	5	148087	167.458	1
23	22334		Juan		Serrano		1646 Seal Way		Burien	Washing	United St	98168	780-555-(	5/24/1966	Graduate	Skilled M	M	M	1	0	0	0	112248	84.8383	0
24	26530		Riley	R	Morgan		2827 Red Leaf		Lake Osv	Oregon	United St	97034	660-555-(	3/13/1969	Graduate	Professic	F	M	0	0	0	5	94217	55.2234	0
25	15411		Jay		Chandra		2926 Woodside Co		Burlingar	Californi	United St	94010	146-555-0	#######	Graduate	Manager	M	S	1	2	0	4	122711	56.1509	0
26	24792		Clifford	G	Malhotra		3356 Eastgate Ave.		Colma	Californi	United St	94014	894-555-(	10/17/1970	Bachelor	Skilled M	M	M	1	1	1	1	58716	87.625	0
27	18812		Kyle	E	Adams		9884 Pacifica Ave.		Victoria	British Co	Canada	V8V	274-555-(	#######	Partial Co	Professic	M	S	1	2	3	4	128140	122.344	1
28	27688		Arthur		Sanz		2707 Virgil Street		Goulburr	New Sou	Australia	2580	1 (11) 500	6/28/1963	Bachelor	Professic	M	S	0	1	0	0	101320	76.5004	1
29	14513		Alexandra		Turner		3995 Tiffin Dr.		Burlingar	Californi	United St	94010	205-555-(	11/20/1968	Graduate	Manager	F	M	1	3	0	1	150227	60.3288	0
30	28847		Carly	G	Xu		2206 Clear View Ci		Concord	Californi	United St	94519	270-555-(	5/23/1968	Bachelor	Skilled M	F	S	1	1	2	2	85726	66.0248	1
31	11168		David		Rodriguez		38 Shangri-la Rd.		Oakland	Californi	United St	94611	730-555-(	5/15/1968	Bachelor	Manager	M	S	0	1	0	0	125452	80.0599	1

Figure 4.8

Conclusion

The result shows machine learning approach can accurately predict bike buyers and average monthly spending of a customer with an accuracy of 77.77% to predict bike buyers and 98.7% to predict average monthly spending of a customer. Mean ROC AUC score is 85.59% which means ability of the model to predict more priority class is 85.59% more than less priority class. In this project more priority class is 1(buyers) that means company are more interested on customers who are more likely to buy. Company wants the good customers to be predicting accurately and not wants to leave a single good customer, ironically a bad customer misclassified as good customer is of little harm. In a survey it has been found that maintaining a good relation with only good customers instead of all can increase profit to a much extent. Now company also know how much a customer is going to spend on monthly basis by this knowledge it can offered a customer the best product which falls in their budgets instead of offering each and every product which guaranteed customer satisfaction.

Reference

https://en.m.wikipedia.org/wiki/Customer_analytics#
https://towardsdatascience.com/5-types-of-regression-and-their-properties-c5e1fa12d55e
https://en.wikipedia.org/wiki/Regression_analysis
https://seaborn.pydata.org/generated/seaborn.violinplot.html
http://www.scikit-learn.org/

Appendix A

Train code

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

from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split,KFold,GridSearchCV,cross_val_score
from sklearn.metrics import
accuracy_score,confusion_matrix,r2_score,mean_squared_error,precision_recall_curve,f1_score,auc,roc_auc_
score,roc_curve,make_scorer
from sklearn.externals import joblib
np.random.seed(9)

train_data=pd.read_csv('AdvWorksCusts.csv')
print('\t\t\tCustomer Spending Beheviour Prediction\n\n\t\tTraining Data\n',train_data.head(),'\n\n\t\tData
Information\n')
train_data.info()

label_data1=pd.read_csv('AW_AveMonthSpend.csv')
label_data2=pd.read_csv('AW_BikeBuyer.csv')

train_data.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
label_data1.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
label_data2.drop_duplicates(subset='CustomerID',keep='last',inplace=True)

label1=label_data1['AveMonthSpend']
label2=label_data2['BikeBuyer']

train_data['age']=(train_data['BirthDate'].apply(lambda x:x[:4])).apply(lambda x:2019-np.int64(x))

train_data.drop(columns=['CustomerID','Title','FirstName','MiddleName','LastName','City','StateProvinceName
','Suffix','AddressLine2','AddressLine1','PostalCode','PhoneNumber','BirthDate'],inplace=True)

num_col=['YearlyIncome','age']
```

```
cat_col=['CountryRegionName','Education','Occupation','Gender','MaritalStatus','HomeOwnerFlag','NumberCa
rsOwned','NumberChildrenAtHome','TotalChildren']
def vizual(data,columns,label=label2,type='bar'):
    if type=='bar':
        for col in columns:
            sns.countplot(y=data[col])
            plt.show()
    if type=='hist':
        for col in columns:
            sns.distplot(data[col],bins=30)
            plt.show()
    if type=='scatter':
        for col in columns:
            sns.scatterplot(col,label,data=data,s=10)
            plt.show()
    if type=='violin':
        for col in columns:
            if(data[col].dtype=='object'):
                sns.violinplot(col,label,data=data)
                plt.show()
            else:
                sns.violinplot(label,col,data=data)
                plt.show()

print('\n\t\tData Vizualization\n\n\tBar Plot')
vizual(train_data,cat_col,type='bar',label=label2[label2==1])
print('\tHistogram Plot')
vizual(train_data,num_col,type='hist',label=label2[label2==1])
print('\tScatter Plot')
vizual(train_data,num_col,type='scatter',label=label1)
print('\tViolin Plot')
vizual(train_data,train_data.columns,type='violin')

def scale(data,columns,type='min_max'):
    for col in columns:
        if(type=='z score'):
            data[col]=(data[col]-np.mean(data[col]))/(np.std(data[col]))
        if(type=='min_max'):
            data[col]=(data[col]-np.min(data[col]))/(np.max(data[col])-np.min(data[col]))
        if(type=='log'):
            data[col]=np.log(data[col])
        if(type=='inverse'):
            data[col]=(1/data[col])
    return data
```

```
def preperation(data):
    data['Gender'].replace({'M':0.9,'F':0.1},inplace=True)
    data['NumberChildrenAtHome'].replace({0:0.1,1:0.3,2:0.5,3:0.7,4:0.9,5:1.1},inplace=True)

    data['s_age']=data['Gender']*data['YearlyIncome']/data['age']*data['NumberChildrenAtHome']
    data['old']=data['age'].apply(lambda x:0 if x<=60 else 1)
    data['gnch']=data['Gender']*data['NumberChildrenAtHome']

    data['NumberChildrenAtHome'].replace({1:0,2:1,3:1,4:1,5:1},inplace=True)
    data['NumberCarsOwned'].replace({1:0,2:0,3:1,4:1},inplace=True)
    data['TotalChildren'].replace({1:0,2:0,3:1,4:1,5:1},inplace=True)

    data=pd.get_dummies(data)
    data=scale(data,columns=['s_age'],type='log')
    data=scale(data,data.select_dtypes('int64','float64'))

    return data

train_data=preperation(train_data)

print('\n\t\tData Vizualization After Data Modification\n\n\tHistogram Plot')
vizual(train_data,['s_age'],type='hist')
print('\tScatter Plot')
vizual(train_data,['s_age'],type='scatter',label=label1)
print('\tBar Plot')
vizual(train_data,['gnch','old'],type='bar')
print('\tViolin Plot')
vizual(train_data,['gnch','old'],type='violin')
def regression(x,y=label1,test_size=0.2):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=test_size)
    model=make_pipeline(PolynomialFeatures(degree=2),LinearRegression())
    model.fit(x_train,y_train)
    return model,x_test,y_test

def classify(x,y=label2,test_size=0.2):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=test_size)

model=make_pipeline(PolynomialFeatures(degree=2),LogisticRegression(C=0.7,penalty='l1',class_weight='bala
nced',solver='liblinear'))
    model.fit(x_train,y_train)
    return model,x_test,y_test
def roc_pr_f1(model,x,y,show=False,plot=False,show_all=False):
    no_skill_probs=[0 for i in range(len(y))]
```

```
    lr_probs=model.predict_proba(x)[:,1]
    pred=model.predict(x)

    ns_fpr,ns_tpr,_=roc_curve(y,no_skill_probs)
    lr_fpr,lr_tpr,_=roc_curve(y,lr_probs)

    ns_precision,ns_recall,_=precision_recall_curve(y,no_skill_probs)
    lr_precision,lr_recall,_=precision_recall_curve(y,lr_probs)
    ns_f1,ns_pr_auc,ns_roc_auc=0.0,auc(ns_recall,ns_precision),roc_auc_score(y,no_skill_probs)
    lr_f1,lr_pr_auc,lr_roc_auc=f1_score(y,pred),auc(lr_recall,lr_precision),roc_auc_score(y,lr_probs)

    if show or show_all:
        print('Accuracy score=',accuracy_score(y,pred))
        print('No Skill: F1 Score=',ns_f1,'\t\tPR AUC=',ns_pr_auc,' ROC AUC=',ns_roc_auc)
        print('Logistic: F1 Score=',lr_f1,' PR AUC=',lr_pr_auc,' ROC AUC=',lr_roc_auc)

    if plot or show_all:
        print('\n\tPrecision VS Recall')
        plt.plot(ns_recall,ns_precision,marker='.',label='No_Skill')
        plt.plot(lr_recall,lr_precision,marker='.',label='Logistics')
        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.legend()
        plt.show()

        print('\n\tROC Curve')
        plt.plot(ns_fpr,ns_tpr,marker='.',label='No_Skill')
        plt.plot(lr_fpr,lr_tpr,marker='.',label='Logistics')
        plt.xlabel('False positive rate')
        plt.ylabel('True positive rate')
        plt.legend()
        plt.show()

    return lr_roc_auc,lr_pr_auc,lr_f1

def conf_mat(y_true,y_pred):
    cfm = confusion_matrix(y_true,y_pred)
    sns.heatmap(cfm, annot=True)
    print('\n\tConfusion Matrix')
    plt.xlabel('Predicted classes')
    plt.ylabel('Actual classes')
    plt.show()

def evaluate(model,x_test,y_test,type='class',conf=True,show=False,plot=False,show_all=False):
```

```
    if type=='class':
        pred=model.predict(x_test)
        roc_auc,pr_auc,f1=roc_pr_f1(model,x_test,y_test,show,plot,show_all)
        if conf or show_all:
            conf_mat(y_test.values,pred)

        return (1-roc_auc)

    if type=='reg':
        pred=model.predict(x_test)
        rmse=np.sqrt(mean_squared_error(y_test,pred))

        if show or show_all:
            print('R2 score = ',r2_score(y_test,pred),'\nRoot Mean Squared Error=',rmse)

        if plot or show_all:
            error=pred-y_test
            plt.scatter(y_test,error,s=1)
            plt.xlabel('Actual value')
            plt.ylabel('Error')
            plt.show()

            error.hist(figsize=(5,5),bins=30)
            plt.xlabel('Error')
            plt.ylabel('Count')
            plt.show()

            plt.scatter(y_test,pred,s=1)
            plt.xlabel('Actual value')
            plt.ylabel('Predicted Value')
            plt.show()

        return rmse
def cv(model,data,label,scoring='roc_auc',fold=5):
    kfold=KFold(n_splits=fold,shuffle=True,random_state=6)
    score=cross_val_score(model,data,label,scoring=scoring,cv=kfold)
    return (score.mean())

def nested_cv(model,data,label,scoring='roc_auc',fold=10):
     inside=KFold(n_splits=fold,shuffle=True,random_state=2)
     outside=KFold(n_splits=fold,shuffle=True,random_state=7)

clf=GridSearchCV(model,param_grid={"C":[0.1,0.7,1,10]},cv=inside,scoring=make_scorer(roc_auc_score),retur
n_train_score=False)
```

```
        clf.fit(data,label)
        score=cross_val_score(clf,data,label,scoring=scoring,cv=outside)
        return clf.best_estimator_.C,(score.mean())
def predict(model,x_test,y_test):
    predicted=pd.DataFrame()
    predicted['Actua Value']=y_test
    predicted['Predicted Value']=model.predict(x_test)
    return predicted
reg_model,x_test,y_test=regression(train_data)
print('\n\t\tPrediction on Validation set\n\tAvarage Monthly Speding of
Customers\n',predict(reg_model,x_test,y_test),'\n\n\tEvaluation\n')
evaluate(reg_model,x_test,y_test,type='reg',show_all=True)
joblib.dump(reg_model,'reg_model.sav')
cv_score=cv(reg_model,train_data,label=label1,scoring='r2')
print('\nMean R2 score after cross validation = ',cv_score)
class_model,x_test,y_test=classify(train_data,label2)
print('\n\tBike Buying Behaviour of Customers\n',predict(class_model,x_test,y_test),'\n\n\tEvaluation\n')
joblib.dump(class_model,'class_model.sav')
evaluate(class_model,x_test,y_test,type='class',show_all=True)
cv_score=cv(class_model,train_data,label=label2,scoring='roc_auc')
print('\nMean ROC AUC after cross validation = ',cv_score)
```

Appendix B

Test code

```
import pandas as pd
import numpy as np
from sklearn.externals import joblib

test_data=pd.read_csv('AW_test.csv')
test_data.drop_duplicates(subset='CustomerID',keep='last',inplace=True)
features=test_data.copy()
features['age']=(features['BirthDate'].apply(lambda x:x[-4:])).apply(lambda x:2019-np.int64(x))
features.drop(columns=['CustomerID','Title','FirstName','MiddleName','LastName','City','StateProvinceName','
Suffix','AddressLine2','AddressLine1','PostalCode','PhoneNumber','BirthDate'],inplace=True)

def scale(data,columns,type='min_max'):
    for col in columns:
        if(type=='z score'):
            data[col]=(data[col]-np.mean(data[col]))/(np.std(data[col]))
        if(type=='min_max'):
            data[col]=(data[col]-np.min(data[col]))/(np.max(data[col])-np.min(data[col]))
        if(type=='log'):
            data[col]=np.log(data[col])
        if(type=='inverse'):
            data[col]=(1/data[col])
    return data

def preperation(data):
    data['Gender'].replace({'M':0.9,'F':0.1},inplace=True)
    data['NumberChildrenAtHome'].replace({0:0.1,1:0.3,2:0.5,3:0.7,4:0.9,5:1.1},inplace=True)

    data['s_age']=data['Gender']*data['YearlyIncome']/data['age']*data['NumberChildrenAtHome']
    data['old']=data['age'].apply(lambda x:0 if x<=60 else 1)
    data['gnch']=data['Gender']*data['NumberChildrenAtHome']

    data['NumberChildrenAtHome'].replace({1:0,2:1,3:1,4:1,5:1},inplace=True)
    data['NumberCarsOwned'].replace({1:0,2:0,3:1,4:1},inplace=True)
    data['TotalChildren'].replace({1:0,2:0,3:1,4:1,5:1},inplace=True)

    data=pd.get_dummies(data)
```

```
        data=scale(data,columns=['s_age'],type='log')
        data=scale(data,data.select_dtypes('int64','float64'))

        return data

features=preperation(features)

reg_model=joblib.load('reg_model.sav')
test_data['Avarage Monthly Spending']=reg_model.predict(features)

class_model=joblib.load('class_model.sav')
test_data['Bike Buyer']=class_model.predict(features)

test_data.to_csv('Result.csv')
```