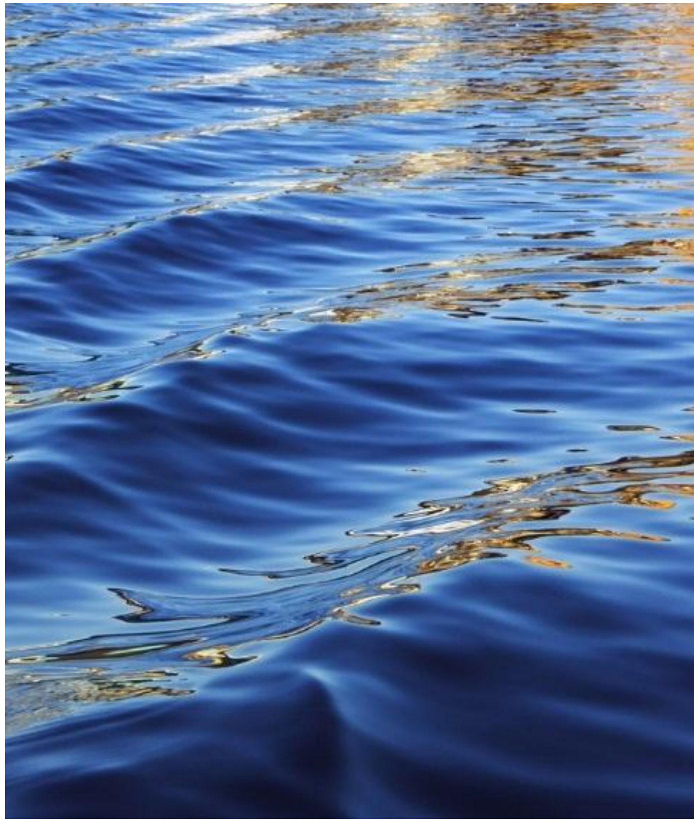
Analyzing Bike Buyers

Jungju Lim



Agenda

1. Case Study Introduction
2. Data Preparation
   - ✔ Data Checking
   - ✔ Data Cleansing
   - ✔ Data Exploration
3. Segmentation (Cluster Analysis)
4. Buyer Propensity (Predictive Analytics/ML)
5. Conclusion

Bike Buyers Case Study - Introduction

Goals

- **Segmentation:** Identify and group customers with similar characteristics for targeting
- **Buyer Propensity:** Create predictive models to predict bike buying propensity

Case Data Set

This dataset has details of 1000 users from different backgrounds and whether they buy a bike or not

Features

ID, Marital Status, Gender, Income, Children, Education, Occupation, Homeowner, Cars, Commute Distance, Region, Age, Purchased Bike

	ID	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age	Purchased Bike
0	12496	Married	Female	40000.0	1.0	Bachelors	Skilled Manual	Yes	0.0	0-1 Miles	Europe	42.0	No
1	24107	Married	Male	30000.0	3.0	Partial College	Clerical	Yes	1.0	0-1 Miles	Europe	43.0	No
2	14177	Married	Male	80000.0	5.0	Partial College	Professional	No	2.0	2-5 Miles	Europe	60.0	No
3	24381	Single	NaN	70000.0	0.0	Bachelors	Professional	Yes	1.0	5-10 Miles	Pacific	41.0	Yes
4	25597	Single	Male	30000.0	0.0	Bachelors	Clerical	No	0.0	0-1 Miles	Europe	36.0	Yes

Segmentation and Buyer Propensity

Cluster Analysis and Predictive Analytics using Machine Learning

Cluster analysis in machine learning (Segmentation)

- Clustering is an unsupervised machine learning method of identifying and grouping similar data points in large datasets without concern for the specific outcome.

Predictive analytics in machine learning (Buyer Propensity)

- In contrast with Clustering, Predictive analytics uses a supervised machine learning method. Machine learning uses advanced mathematics to find patterns in datasets and creates an optimal model that has minimal differences from output labels.

Why do we use machine learning?

- Machine learning is a powerful way for companies to get value from the massive amounts of data and to generate the results quickly and efficiently.

Segmentation and Buyer Propensity

Steps for Modeling Process

I. **Data Preparation**

- ✔ Data Checking
- ✔ Data Cleansing
- ✔ Data Exploration

II. **Segmentation (Cluster Analysis)**

III. **Buyer Propensity (Predictive Analytics)**

Data Preparation

Data Checking

Data Count, Data Types, Missing Values

```
# brief information of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                1000 non-null   int64
 1   Marital Status    993 non-null    object
 2   Gender            989 non-null    object
 3   Income            994 non-null    float64
 4   Children          992 non-null    float64
 5   Education         1000 non-null   object
 6   Occupation        1000 non-null   object
 7   Home Owner        996 non-null    object
 8   Cars              991 non-null    float64
 9   Commute Distance  1000 non-null   object
 10  Region            1000 non-null   object
 11  Age               992 non-null    float64
 12  Purchased Bike    1000 non-null   object
dtypes: float64(4), int64(1), object(8)
memory usage: 101.7+ KB
```

```
# brief statistic infomation
df.describe()
```

	ID	Income	Children	Cars	Age
count	1000.000000	994.000000	992.000000	991.000000	992.000000
mean	19965.992000	56267.605634	1.910282	1.455096	44.181452
std	5347.333948	31067.817462	1.626910	1.121755	11.362007
min	11000.000000	10000.000000	0.000000	0.000000	25.000000
25%	15290.750000	30000.000000	0.000000	1.000000	35.000000
50%	19744.000000	60000.000000	2.000000	1.000000	43.000000
75%	24470.750000	70000.000000	3.000000	2.000000	52.000000
max	29447.000000	170000.000000	5.000000	4.000000	89.000000

```
# check the number of null data
df.isna().sum()
```

```
ID                   0
Marital Status       7
Gender              11
Income               6
Children             8
Education            0
Occupation           0
Home Owner           4
Cars                 9
Commute Distance     0
Region               0
Age                  8
Purchased Bike       0
dtype: int64
```

Data Preparation

Data Cleansing

Deal with Missing Values

```
# percentage of null values for each column
(df.isna().sum()[df.isna().sum()>0]/1000)*100
```

```
Marital Status    0.7
Gender            1.1
Income            0.6
Children          0.8
Home Owner        0.4
Cars              0.9
Age               0.8
dtype: float64
```

```
# fill the null values to "Married"
df["Marital Status"].fillna("Married",inplace=True)
```

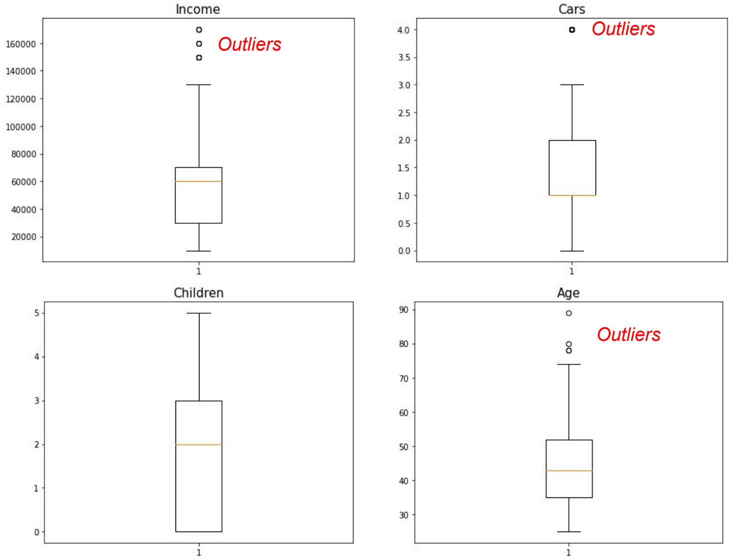
```
# fill the null values to "Yes"
df["Home Owner"].fillna("Yes",inplace=True)
```

```
# fill the null values to "Male"
df["Gender"].fillna("Male",inplace=True)
```

```
# fill the null vales of Income, Children, Cars,
# Age columns to their mean or median value
df.fillna({'Income': df['Income'].mean(),\
          'Children': df['Children'].median(),\
          'Cars': df['Cars'].median(),\
          'Age': df['Age'].median()},inplace=True)
```

Identify Outliers for Removal

	ID	Income	Children	Cars	Age
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	19965.992000	56267.605634	1.911000	1.451000	44.172000
std	5347.333948	30974.380206	1.620403	1.117519	11.316912
min	11000.000000	10000.000000	0.000000	0.000000	25.000000
25%	15290.750000	30000.000000	0.000000	1.000000	35.000000
50%	19744.000000	60000.000000	2.000000	1.000000	43.000000
75%	24470.750000	70000.000000	3.000000	2.000000	52.000000
max	29447.000000	170000.000000	5.000000	4.000000	89.000000



Data Preparation

Data Exploration

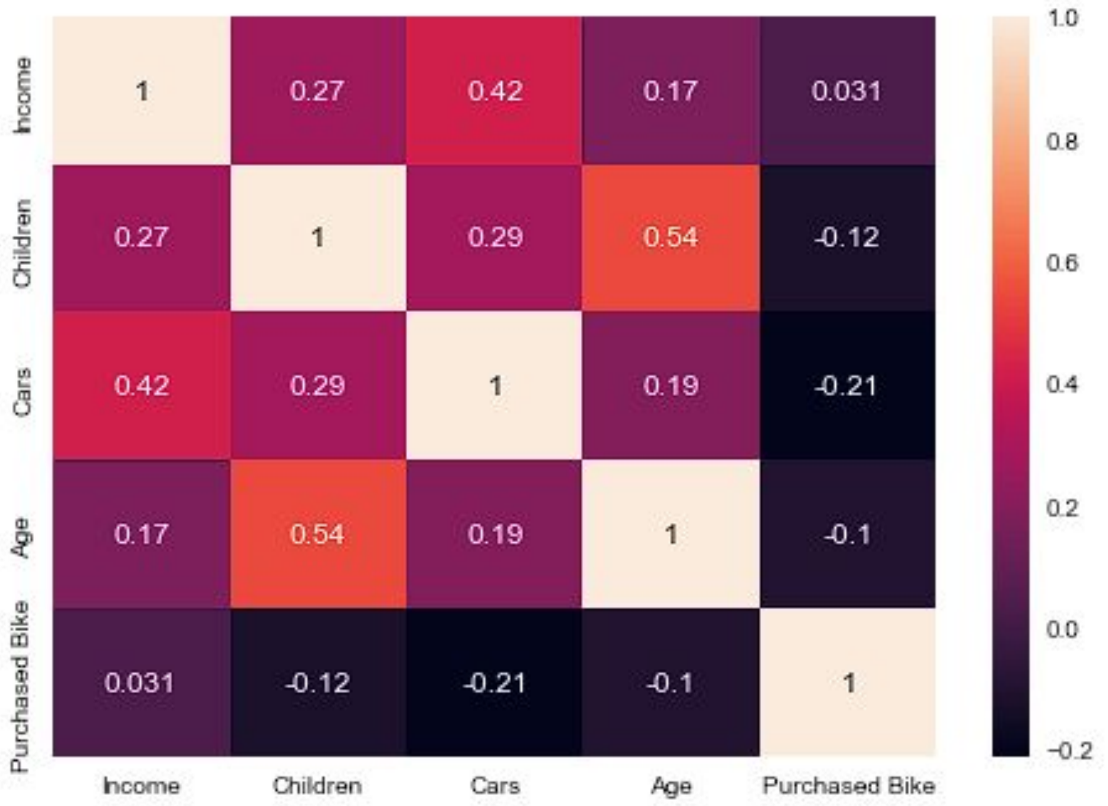
Check for correlation for feature modification

Correlation: to indicate any type of association, in statistics it normally refers to the degree to which a pair of variables are linearly related

```
df.corr()
```

	Income	Children	Cars	Age	Purchased Bike
Income	1.000000	0.267123	0.416835	0.174129	0.031425
Children	0.267123	1.000000	0.285058	0.536773	-0.124675
Cars	0.416835	0.285058	1.000000	0.194397	-0.211386
Age	0.174129	0.536773	0.194397	1.000000	-0.100630
Purchased Bike	0.031425	-0.124675	-0.211386	-0.100630	1.000000

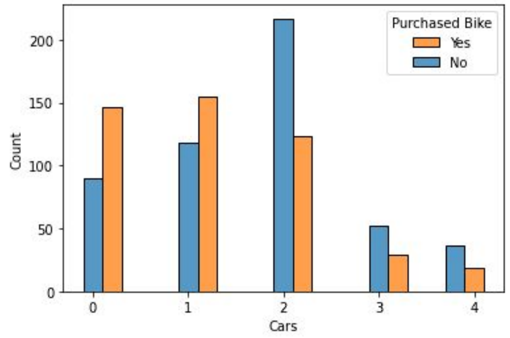
```
ax = sns.heatmap(df.corr(), annot=True)
```



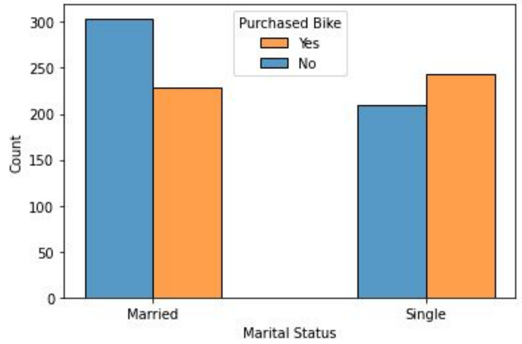
Data Preparation

Data Exploration

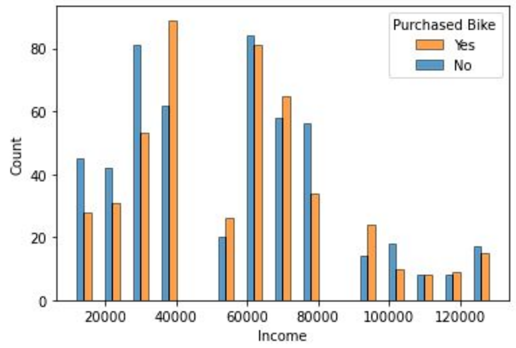
Targets with 0~1 Cars more likely to buy



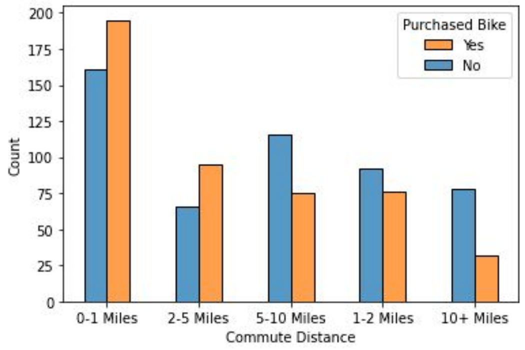
Single are more likely to buy



Mid-level income earners more likely to buy



Shorter commuters more likely to buy



Segmentation

Cluster analysis

Prepare Features (Data) for Modeling

Transform non-numeric values to numeric values

```
# changing all categorical features to numerical features
categorical = []
df_en = df.copy()
for i in df.columns:
    if df[i].dtype == "object":
        categorical.append(i)
for i in categorical:
    df_en[i]= LabelEncoder().fit_transform(df[i])

df_en.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 986 entries, 0 to 999
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Marital Status    986 non-null    int32
 1   Gender            986 non-null    int32
 2   Income            986 non-null    float64
 3   Children          986 non-null    float64
 4   Education         986 non-null    int32
 5   Occupation        986 non-null    int32
 6   Home Owner        986 non-null    int32
 7   Cars              986 non-null    float64
 8   Commute Distance  986 non-null    int32
 9   Region            986 non-null    int32
 10  Age               986 non-null    float64
 11  Purchased Bike    986 non-null    int32
dtypes: float64(4), int32(8)
memory usage: 69.3 KB
```

Ensure Features (Data) avoid Bias

Scaling Features

```
# scaling
scaler = StandardScaler()
scaler.fit(df_en)
df_scale = pd.DataFrame(scaler.transform(df_en), columns=df_en.columns)
df_scale
```

	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Comm Dista
0	-0.921904	-1.020494	-0.520216	-0.559649	-1.202470	1.196085	0.674001	-1.297197	-1.062
1	-0.921904	0.979918	-0.861316	0.676216	1.009536	-1.552959	0.674001	-0.393923	-1.062
2	-0.921904	0.979918	0.844185	1.912082	1.009536	0.508824	-1.483677	0.509351	0.859
3	1.084712	0.979918	0.503085	-1.177582	-1.202470	0.508824	0.674001	-0.393923	1.500
4	1.084712	0.979918	-0.861316	-1.177582	-1.202470	-1.552959	-1.483677	-1.297197	-1.062

Segmentation

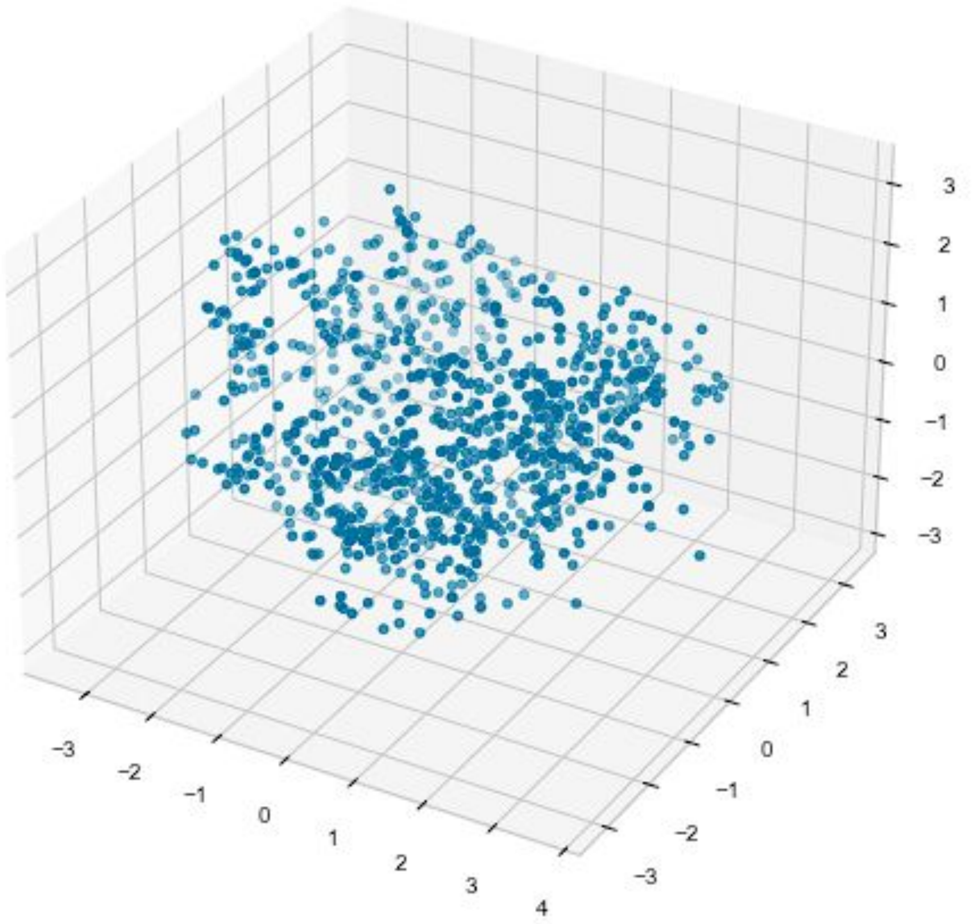
Cluster analysis

Simplify Features (Data) to Improve Interpretation

Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss.

```
# reduce dimentions using PCA
pca = PCA(n_components=3)
pca.fit(df_scale)
PCA_df = pd.DataFrame(pca.transform(df_scale), columns=(["col1","col2", "col3"]))
PCA_df.describe()
```

	col1	col2	col3
count	9.860000e+02	9.860000e+02	9.860000e+02
mean	-9.492069e-17	6.812220e-17	-9.109234e-17
std	1.531138e+00	1.254731e+00	1.185710e+00
min	-3.429656e+00	-3.158967e+00	-2.896069e+00
25%	-1.265473e+00	-8.836928e-01	-9.462069e-01
50%	1.125928e-01	-1.761622e-02	-1.950332e-03
75%	1.270065e+00	9.123375e-01	8.358404e-01
max	3.697362e+00	3.375744e+00	3.197340e+00



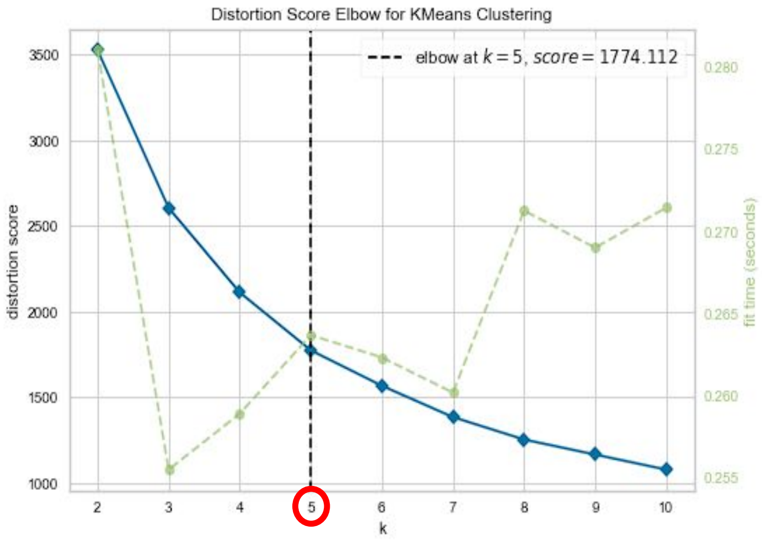
Segmentation

Cluster analysis

Create the optimal number of segments

Elbow method is a way to determine the number of clusters in a data set.

```
# decide the number of clusters to make
k_elbow = KElbowVisualizer(KMeans(), k=10)
k_elbow.fit(PCA_df)
k_elbow.show()
```

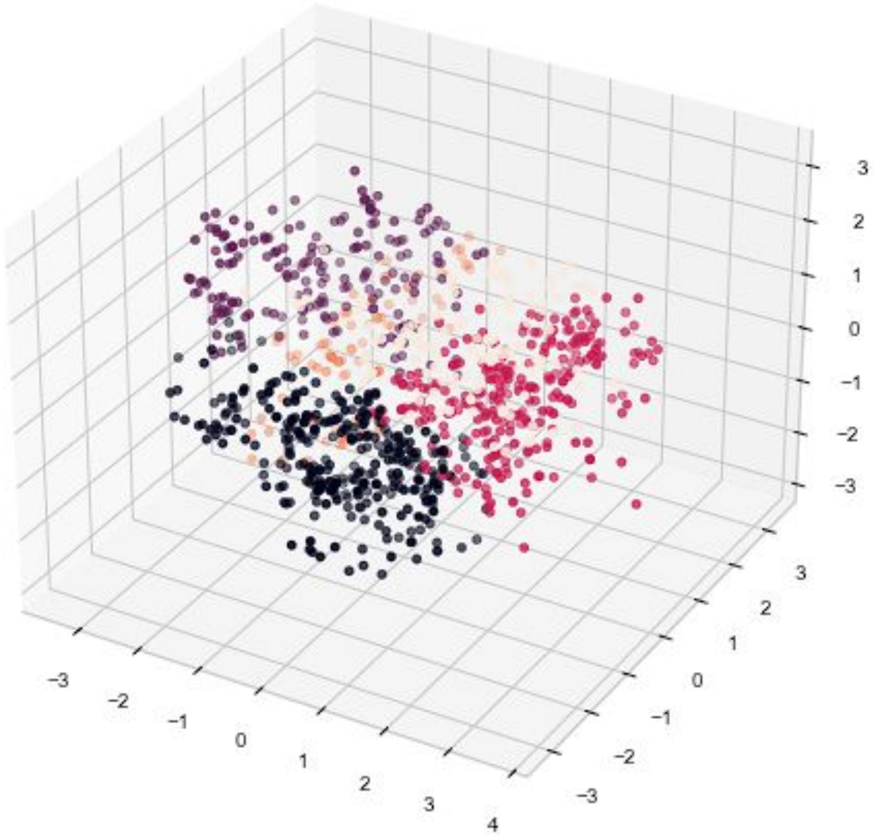


Segmentation

Cluster analysis

Finalize Segments

Leveraging Agglomerative Clustering



	col1	col2	col3	Clusters
0	-1.244564	1.302104	-1.025430	3
1	-0.880557	2.220628	1.286171	1
2	1.882274	0.703037	1.937525	4
3	0.521432	-1.859110	-2.055844	0
4	-3.065150	0.280486	-0.598017	0
...	...	...	...	...
981	1.249371	0.076143	-0.576714	2
982	0.201168	-0.480199	-1.348231	0
983	-0.459614	0.380078	-2.538501	0
984	0.976842	-0.783853	0.613621	4
985	0.890901	-0.278125	-0.129581	2

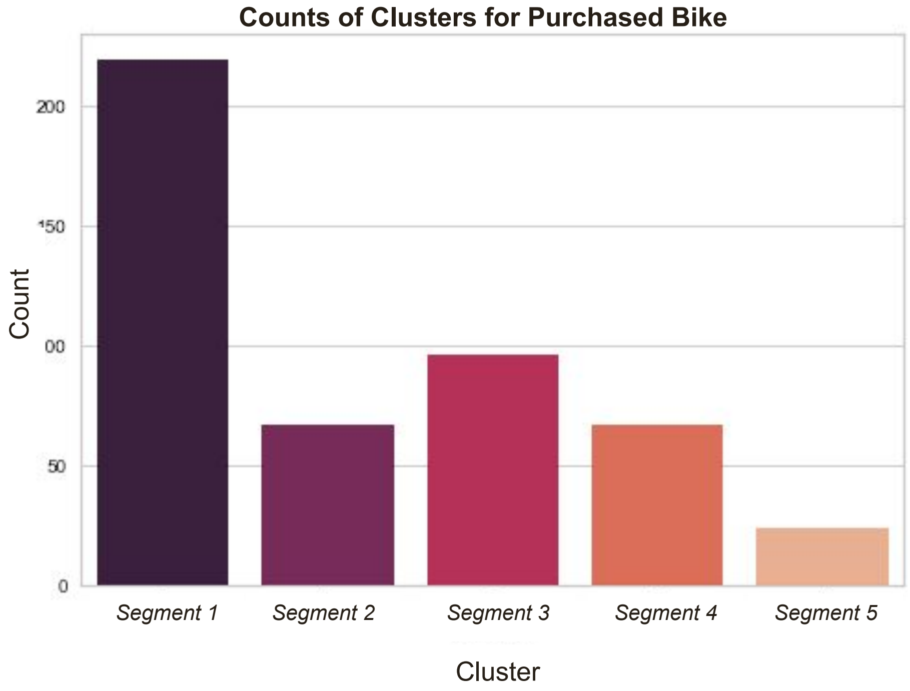
```python
# make a clustering model
AC = AgglomerativeClustering(n_clusters=5)
AC_pred = AC.fit_predict(PCA_df)

PCA_df["Clusters"] = AC_pred
PCA_df
```

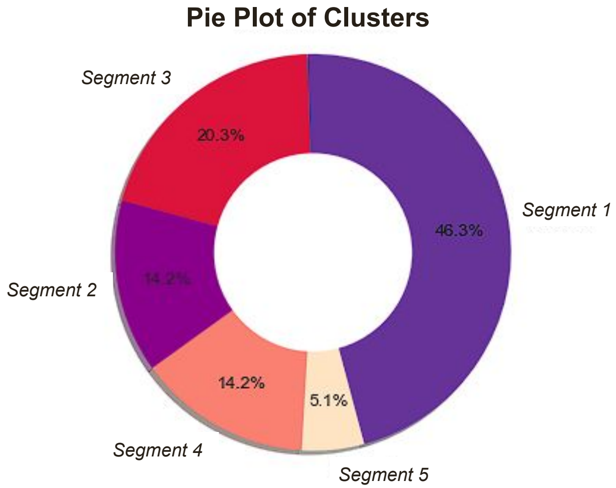
Agglomerative clustering is a hierarchical clustering method. It involves merging examples until the desired number of clusters is achieved.

Segmentation

Identify highest Potential Segments (Clusters)

Segment 1 has the largest number of bike purchasers

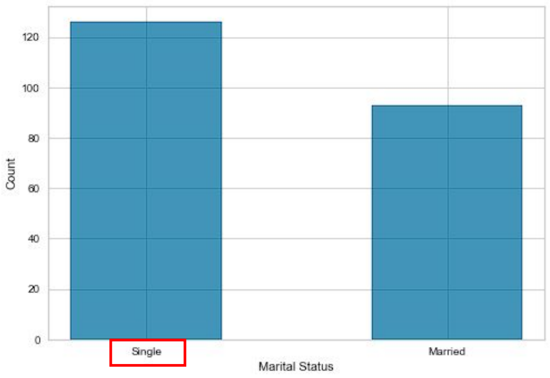




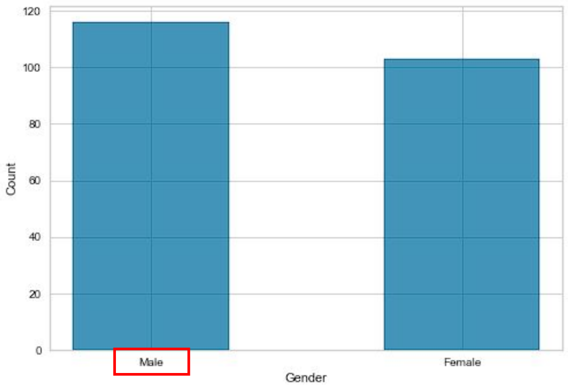
Segmentation

Deep Dive into Segment 1

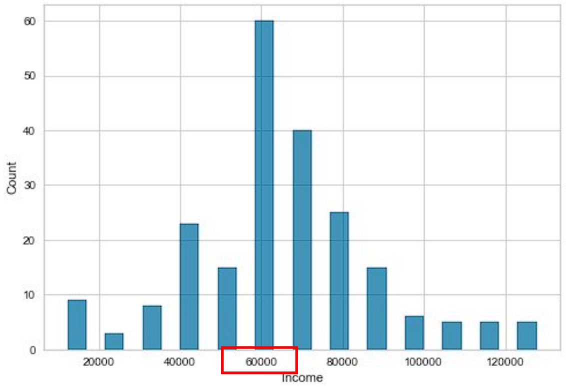
Marital Status: *Single*



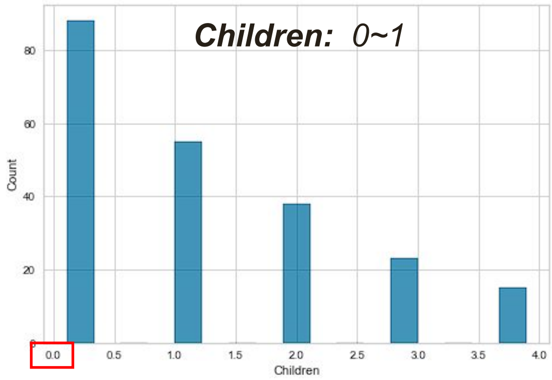
Gender: *Male*



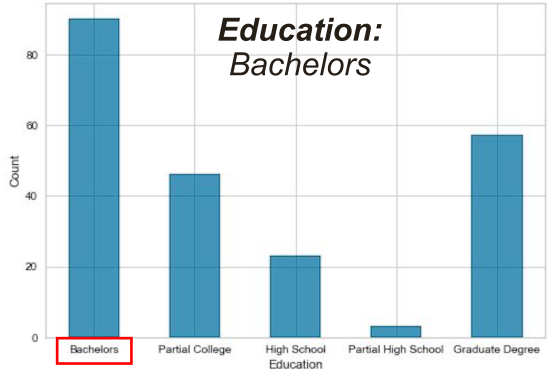
Income: *$60K~$70K*



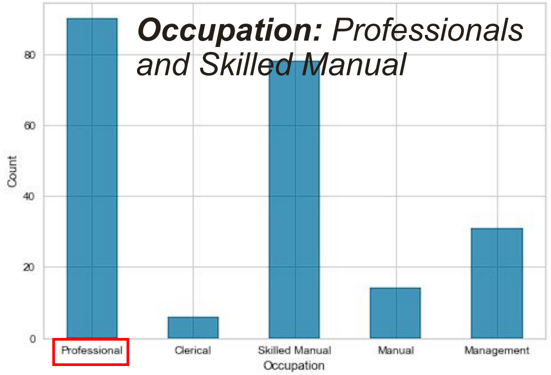
Children: *0~1*



Education: *Bachelors*



Occupation: *Professionals and Skilled Manual*

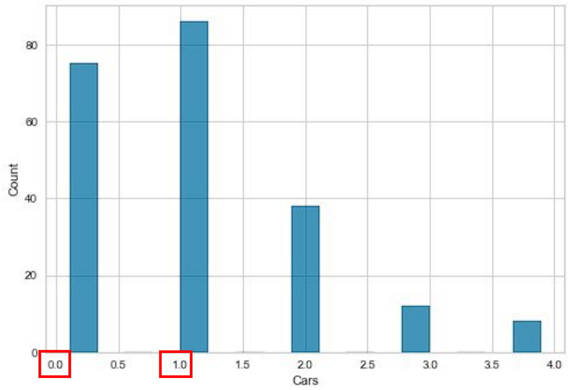


Segmentation

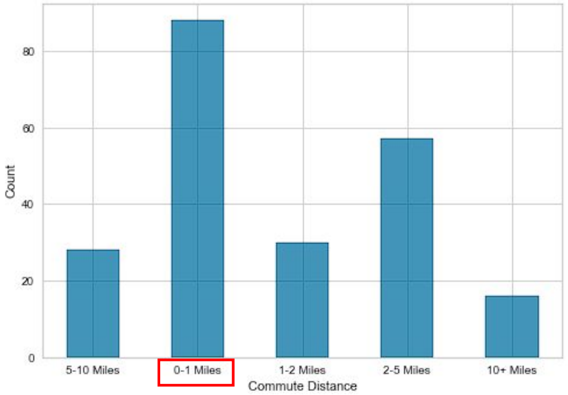
Deep Dive into Segment 1

Home Owner: *Yes*

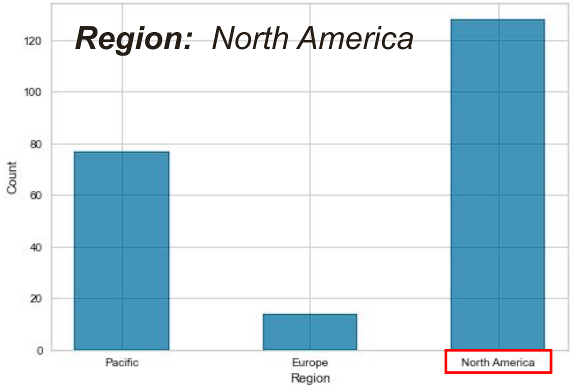
Cars: *0~1*



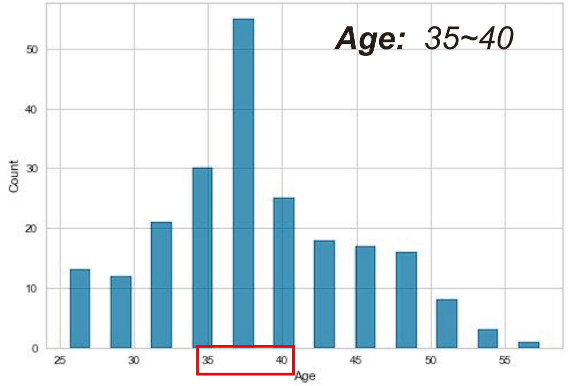
Commute Distance: *0~1 Mile*



Region: *North America*



Age: *35~40*



Segmentation

Summary Characteristics of Segment 1

Mostly young professional men with no children, that own a home and have a short commute

- ✔ **Income:** 60k to 70k
- ✔ **Children:** 0 to 1
- ✔ **Bachelor's degree**
- ✔ **Occupation:** professional & skilled manual
- ✔ **Home owner**
- ✔ **Cars:** 0 to 1
- ✔ **Commute distance:** 0 to 1 mile
- ✔ **North America**
- ✔ **Age:** 35 to 40

Buyer Propensity Modeling

Regression analysis (Predictive Analytics/ML)

Split data into the train data set and the test data set

```
# split the train set and the test set
train, test = train_test_split(df_enc,test_size=0.2, random_state=0)
```

```
train.shape, test.shape
```

```
((788, 29), (198, 29))
```

```
# define input(X) and output(Y) for the train set and the test set
train_X = train.drop(columns="Purchased Bike")
train_Y = train["Purchased Bike"]
test_X = test.drop(columns="Purchased Bike")
test_Y = test["Purchased Bike"]
```

Buyer Propensity Modeling

Regression analysis (Predictive Analytics/ML)

Predictive Model Development

```
# fit model using XGBoost
model = XGBClassifier(n_estimators=90,max_depth=3)
model.fit(train_X, train_Y)
```

```
# prediction
pred = model.predict(test_X)
pred
```

```
array([0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1])
```

XGBoost, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems

Buyer Propensity Modeling

Regression analysis (Predictive Analytics/ML)

Model Testing

```
# compare the real Y values to predictions
output = pd.DataFrame(list(test_Y),columns=["Y"])
output["Prediction"] = list(pred)
output.head(20)
```

	Y	Prediction
0	1	0
1	0	0
2	1	0
3	1	0
4	1	1
5	1	1
6	1	1
7	1	0
8	0	0
9	1	1
10	1	0
11	1	1
12	1	1
13	0	0
14	0	1
15	1	1
16	1	0
17	0	0
18	1	0
19	1	0

```
# check the accuracy of predictions
accuracy = accuracy_score(test_Y,pred)
print("{}%".format(round(accuracy*100,2)))
```

```
67.17%
```

Conclusion

Modeling Results

Developed a predictive model that can be run against target audiences to predict potential targets with a higher propensity to buy a bike. (Prioritization of target lists)

Increase targeting efficiency to improve marketing performance and grow revenue

Verify Model – Based on Segment 1 Data

```
target_X.loc[986:987].reset_index()
```

	index	Marital Status	Gender	Income	Children	Education	Occupation	Home Owner	Cars	Commute Distance	Region	Age
0	986	1	1	65000.0	0.0	0	3	1	0	0	1	36.0

```
pred_target = rf.predict(target_X.loc[986:987])
pred_target
```

```
C:\Users\Lesson6\anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but RandomForestClassifier
was fitted without feature names
  warnings.warn(
```

```
array([1])
```