Table of the content

Ads Click-Through Rate Prediction

1. Project Description:

Abstract:

The goal of this project is to build a predictive model that can estimate the click-through rate of advertisements. By analyzing various factors such as ad content, user demographics, and historical data, the system will provide advertisers with insights into the likelihood of users clicking on their ads. The predictive model will be trained using machine learning techniques to learn patterns from past data and make predictions for future ads.

1.1 Introduction:

In the modern era of online advertising, it is crucial for advertisers to understand the effectiveness of their ads and optimize their campaigns accordingly. Predicting the click-through rate of ads can help advertisers make informed decisions about ad placement, targeting, and budget allocation. By accurately estimating the CTR, advertisers can improve their ad performance and achieve higher conversion rates.

1.2 Existing System:

Currently, many advertisers rely on basic metrics such as impressions and clicks to evaluate the performance of their ads. However, these metrics provide limited insights into the overall effectiveness of an ad campaign. Predictive models for CTR estimation are becoming increasingly popular as they offer a more comprehensive analysis of ad performance. Existing systems use machine learning algorithms to analyze historical data and generate predictions based on various features such as ad content, user behavior, and contextual information.

1.3 Proposed System:

The proposed system will leverage machine learning techniques to develop a predictive model for estimating the click-through rate of ads. It will incorporate a range of features such as ad characteristics (e.g., ad format, text, images), user demographics (e.g., age, gender, location), and contextual information (e.g., time of day, device type) to make accurate predictions. The system will be trained on a large dataset of historical ad impressions and click data to learn patterns and relationships between features and the CTR. The trained model will then be used to predict the CTR for new ads in real-time.

1.4 Software Requirements:

- Programming language: python
- Data analysis and machine learning libraries: scikit-learn,pandas,
- User interface design tools:Google colab

1.5 Hardware Requirements:

- Processor: Intel Core i5
- RAM: 8 GB or more
- Storage: 256 GB

- Stable internet connection
- Operating System: Windows, Linux

2.Logical Development

To predict click-through rate (CTR) for ads, several logical steps can be followed:

1. Data Collection: Gather a dataset that includes historical information about ads, such as impressions and clicks, as well as relevant features like ad text, ad placement, device type, user demographics, and contextual information.

2. Data Preprocessing: Clean the collected data by handling missing values, removing duplicates, and normalizing or scaling numerical features. Encode categorical variables using techniques like one-hot encoding or label encoding.

3. Feature Engineering: Extract relevant features from the raw data that could potentially impact the CTR. For example, create new features like the length of the ad text, the presence of specific keywords, or the interaction between different variables.

4. Splitting the Data: Divide the dataset into training and testing sets. The training set will be used to build the CTR prediction model, while the testing set will evaluate its performance.

5. Model Selection: Choose an appropriate machine learning algorithm for CTR prediction. Some commonly used models include logistic regression, decision trees, random forests, gradient boosting, or neural networks. Consider factors like interpretability, computational efficiency, and the ability to handle high-dimensional and sparse data.

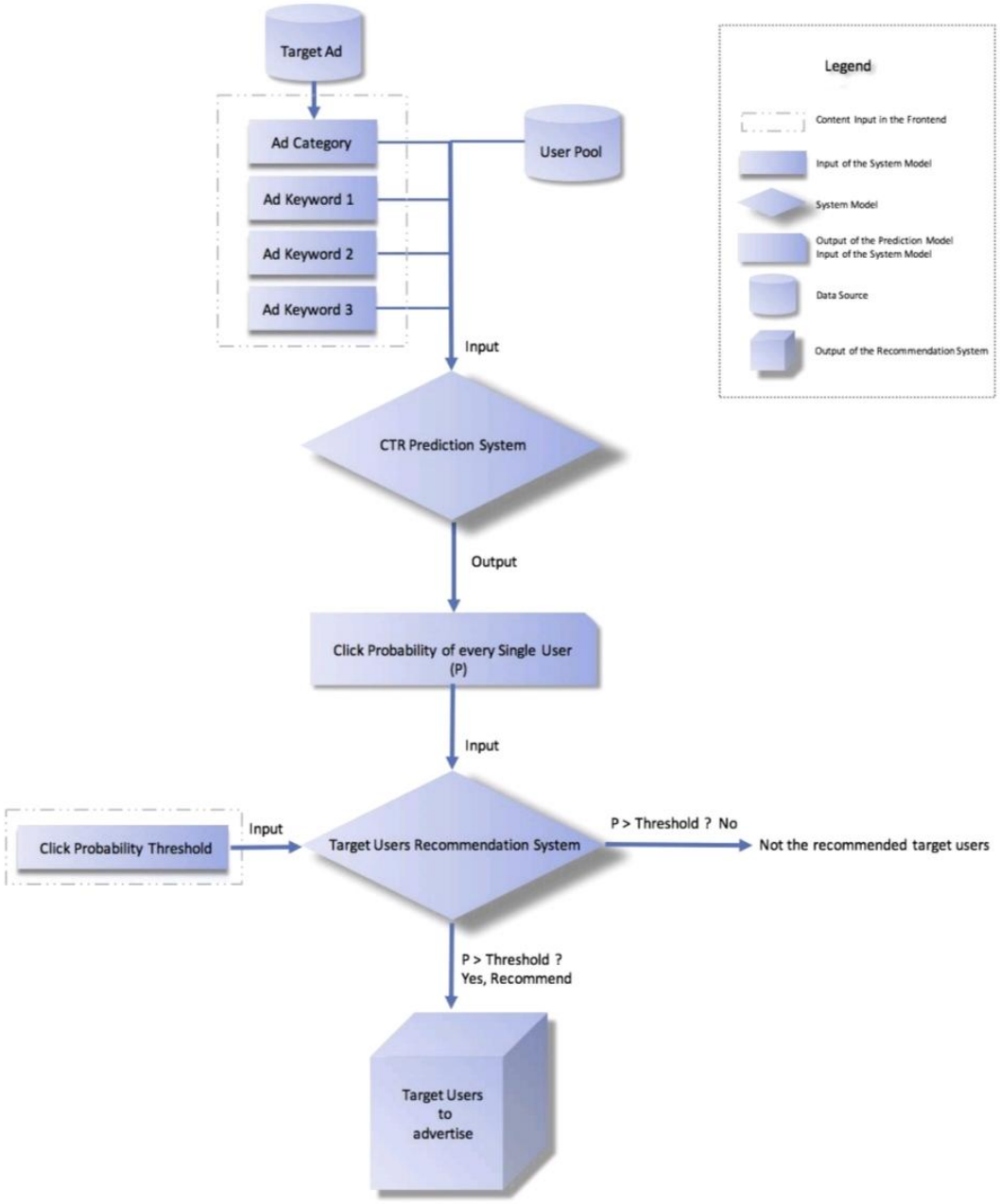
6. Model Training: Fit the selected model to the training data. This process involves adjusting the model's parameters to minimize the difference between the predicted CTR and the actual CTR values in the training set. Techniques like cross-validation or grid search can be employed to optimize the model's hyperparameters.

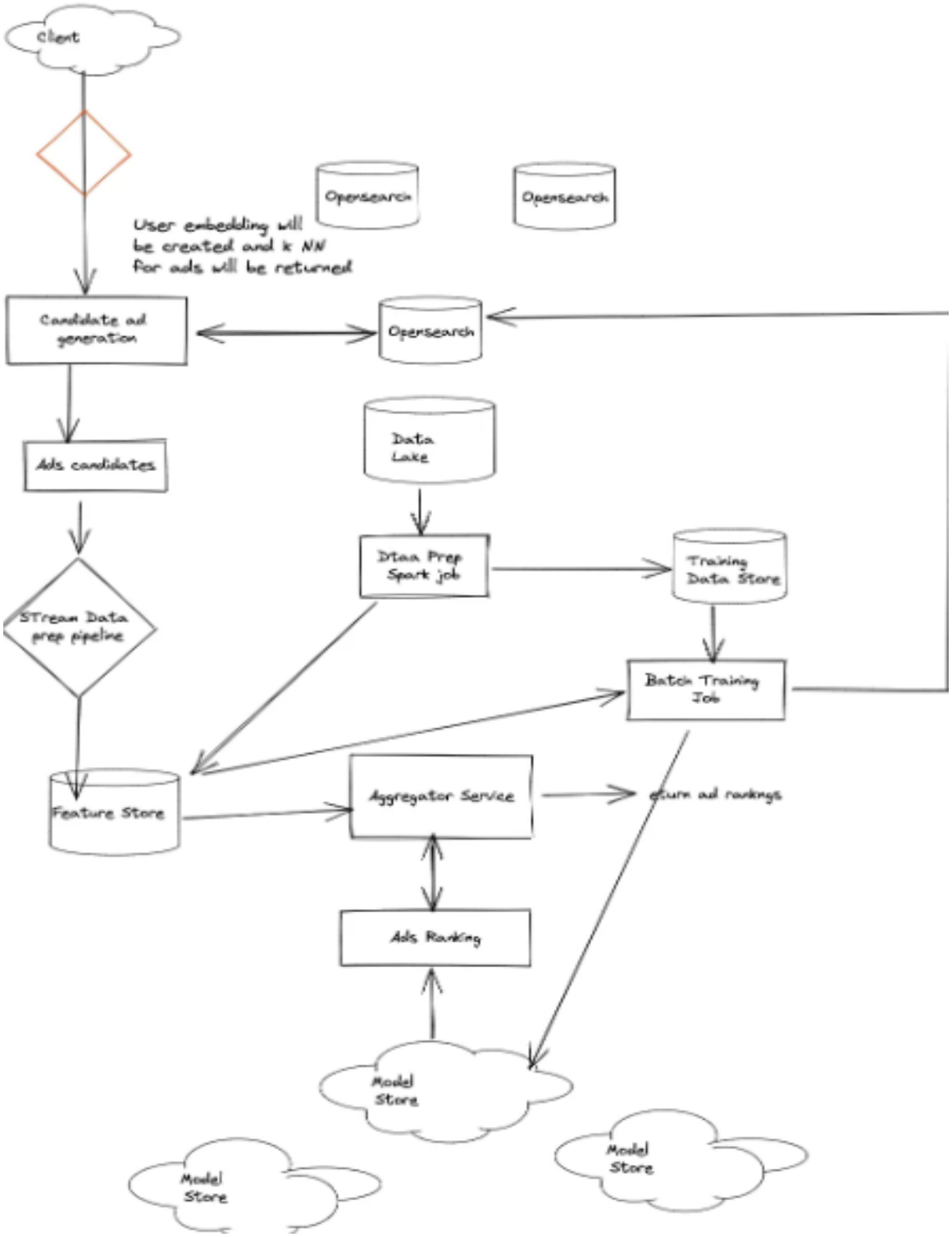
7. Model Evaluation: Assess the performance of the trained model using evaluation metrics such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic curve (AUC-ROC). Compare the model's performance on the testing set against baseline or industry benchmarks.

8. Model Deployment: Once the model is deemed satisfactory, deploy it to predict the CTR for new ads. This can be done by integrating the model into an existing ad-serving system or creating a separate application or API that accepts ad-related information as input and outputs the predicted CTR.

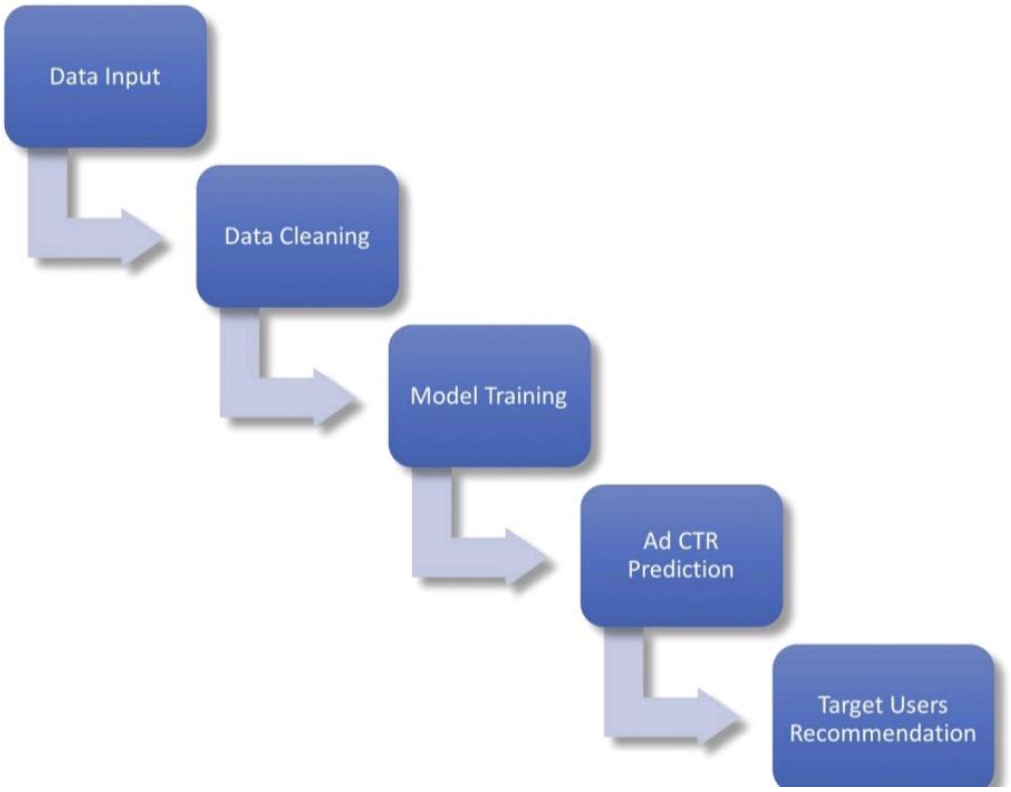
9. Monitoring and Iteration: Continuously monitor the model's performance in real-world scenarios and collect new data to further refine the model. Revisit feature engineering or try different algorithms to improve accuracy and adapt to changing user behavior or market conditions.

2.1 Architectural diagram




Client
Opersearch
Opersearch
User embedding will
be created and k NN
For ads will be returned
Candidate ad
generation
Opersearch
Ads candidates
Data
Lake
Dtaa Prep
Spark job
Training
Data Store
STream Data
prep pipeline
Batch Training
Job
Feature Store
Aggregator Service
return ad rankings
Ads Ranking
Model
Store
Model
Store
Model
Store

2.2 Dataflow diagram



4.Program Design

5.Testing

Test Case 1:

- User demographics: Age: 40, Gender: Male, Location: San Francisco
- Browsing history: Recently visited technology news websites
- Contextual information: Currently viewing an article about the latest smartphones
- Expected result: Moderate probability of clicking on an advertisement for a new smartphone model or a technology gadget.

Test Case 2:

- User demographics: Age: 18, Gender: Male, Location: London
- Browsing history: Recently visited sports and fitness websites
- Contextual information: Currently viewing a webpage about soccer training drills
- Expected result: Low probability of clicking on an advertisement for a cooking recipe website or a home decor store.

Test Case 3:

- User demographics: Age: 35, Gender: Female, Location: Los Angeles
- Browsing history: Recently visited travel and vacation planning websites
- Contextual information: Currently viewing a travel blog post about the best beach destinations

- Expected result: High probability of clicking on an advertisement for a hotel booking website or a vacation package offer.

Test Case 4:

- User demographics: Age: 50, Gender: Male, Location: Chicago
- Browsing history: Recently visited financial news websites
- Contextual information: Currently viewing an article about retirement planning
- Expected result: Moderate probability of clicking on an advertisement for a retirement investment plan or a financial advisory service.

6.Conclusion

logistic regression models have proven to be effective in predicting the probability of user clicks on advertisements or links on a website. By leveraging user demographics, browsing history, and contextual information, these models can analyse relevant features to optimise ad placements and personalise content in real time. Overall, the application of logistic regression models in website click prediction offers significant benefits for businesses seeking to optimise their advertising efforts and deliver a more personalised user experience. By leveraging the power of data analysis and machine learning, companies can make informed decisions to maximise user engagement, increase click-through rates, and ultimately drive conversions and revenue.

7.References

https://www.kaggle.com/c/avazu-ctr-prediction
https://github.com/shubham13p/Ad-Click-Prediction

8.Appendix

8.1 Source Code

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np

Advertise_data = pd.read_csv("ad click data.csv")
Advertise_data.head()

Advertise_data.shape

Advertise_data.info()

Advertise_data.describe()

Advertise_data.isnull().sum()

selectedPredictor=
['Time_Spent','Age','Avg_Income','Internet_Usage','Ad_Topic','City_code', 'Male',
'Time_Period']
FinalData = Advertise_data[selectedPredictor]
FinalData.head()

FinalData['Male'].replace({'Yes':1,'No':0},inplace= True)
FinalData.head()

Final_numData= pd.get_dummies(FinalData)
Final_numData['Clicked']= Advertise_data['Clicked']
Final_numData.head()

Final_numData= pd.get_dummies(FinalData)
Final_numData['Clicked']= Advertise_data['Clicked']
Final_numData.head()
Final_numData.columns

TargetVariable = 'Clicked'
Predictors = ['Time_Spent', 'Age', 'Avg_Income', 'Internet_Usage', 'Male',
      'Ad_Topic_product_1', 'Ad_Topic_product_10', 'Ad_Topic_product_11',
      'Ad_Topic_product_12', 'Ad_Topic_product_13', 'Ad_Topic_product_14',
      'Ad_Topic_product_15', 'Ad_Topic_product_16', 'Ad_Topic_product_17',
      'Ad_Topic_product_18', 'Ad_Topic_product_19', 'Ad_Topic_product_2',
      'Ad_Topic_product_20', 'Ad_Topic_product_21', 'Ad_Topic_product_22',
      'Ad_Topic_product_23', 'Ad_Topic_product_24', 'Ad_Topic_product_25',
      'Ad_Topic_product_26', 'Ad_Topic_product_27', 'Ad_Topic_product_28',
      'Ad_Topic_product_29', 'Ad_Topic_product_3', 'Ad_Topic_product_30',
      'Ad_Topic_product_4', 'Ad_Topic_product_5', 'Ad_Topic_product_6',
      'Ad_Topic_product_7', 'Ad_Topic_product_8', 'Ad_Topic_product_9',
      'City_code_City_1', 'City_code_City_2', 'City_code_City_3',
      'City_code_City_4', 'City_code_City_5', 'City_code_City_6',
      'City_code_City_7', 'City_code_City_8', 'City_code_City_9',
```

```
       'Time_Period_Early-Morning', 'Time_Period_Evening',
       'Time_Period_Mid-Night', 'Time_Period_Morning', 'Time_Period_Night',
       'Time_Period_Noon']

X= Final_numData[Predictors].values
y= Final_numData[TargetVariable].values

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y , test_size = .2 , random_state = 42)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

from sklearn import tree
model= tree.DecisionTreeClassifier()
model.fit(X_train,y_train)


pred_dt =model.predict(X_test)

from sklearn.metrics import confusion_matrix
tab1 = confusion_matrix(pred_dt , y_test)
tab1

tab1.diagonal().sum() / tab1.sum() * 100

model.feature_importances_

from sklearn.metrics import accuracy_score
accuracy_score(y_test, pred_dt)

Advertise_data['Clicked'].hist()

import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x= 'City_code',data= Advertise_data)
plt.show()

def plotbarchart(inpData, Colstoplot):
    %matplotlib inline

    import matplotlib.pyplot as plt


    fig, subplot = plt.subplots(nrows = 1, ncols= len(Colstoplot), figsize = (20,5))
    # OR fig, subplot = plt.subplots(nrows= len(Colstoplot),ncols=1, figsize = (5,15))
    fig.suptitle("bar chart of: " + str(Colstoplot))

    for colnum, plotnum in zip (Colstoplot, range(len(Colstoplot))):
        inpData.groupby(colnum).size().plot(kind='bar',ax= subplot[plotnum])
```

```
plotbarchart(inpData=Advertise_data , Colstoplot= ["Ad_Topic","City_code",
"Male","Time_Period", "Weekday","Month"])

Advertise_data.hist(['Time_Spent','Age','Avg_Income','Internet_Usage','Ad_Topic'],figsize=(
18,10))

ContinuousColList= ['Time_Spent','Age','Avg_Income','Internet_Usage']

import matplotlib.pyplot as plt

fig, subplot = plt.subplots(nrows= 1, ncols= len (ContinuousColList), figsize= (18,5))

for i, plotnum in zip(ContinuousColList, range(len(ContinuousColList))):
    Advertise_data.boxplot(column = i, by= 'Clicked', figsize= (18,5), vert= True,
ax=subplot[plotnum])



CategoricalList= ["Ad_Topic","City_code", "Male", "Time_Period", "Weekday","Month"]

import matplotlib.pyplot as plt

fig, subplot= plt.subplots(nrows= len(CategoricalList),figsize= (15,60))


for colnum, plotnum in zip( CategoricalList, range(len(CategoricalList))):
    crosstabResult= pd.crosstab(index= Advertise_data[colnum], columns=
Advertise_data['Clicked'])
    crosstabResult.plot.bar(ax= subplot[plotnum])
```

8.2 Screen Shot

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
```

```
[ ] Advertise_data = pd.read_csv("ad click data.csv")
    Advertise_data.head()
```

	VistID	Time_Spent	Age	Avg_Income	Internet_Usage	Ad_Topic	Country_Name	City_code	Male	Time_Period	Weekday	Month	Year	Clicked
0	5183153	87.97	43	55901.12	185.46	product_11	Serbia	City_5	No	Mid-Night	Thursday	July	2020	0
1	4023265	51.63	50	39132.00	176.73	product_8	Turkmenistan	City_1	No	Evening	Saturday	June	2020	1
2	4708083	82.37	38	57032.36	210.60	product_6	Northern Mariana Islands	City_2	No	Morning	Tuesday	January	2020	0
3	9771815	62.06	45	48868.00	190.05	product_19	South Africa	City_3	Yes	Morning	Thursday	April	2020	1
4	6451317	77.66	31	61608.23	204.86	product_11	Guadeloupe	City_2	No	Noon	Thursday	January	2020	0

```
[ ] Advertise_data.shape
```

```
(6657, 14)
```

```
Advertise_data.shape
```

```
(6657, 14)
```

```
Advertise_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6657 entries, 0 to 6656
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   VistID          6657 non-null   int64
 1   Time_Spent      6657 non-null   float64
 2   Age             6657 non-null   int64
 3   Avg_Income      6657 non-null   float64
 4   Internet_Usage  6657 non-null   float64
 5   Ad_Topic        6657 non-null   object
 6   Country_Name    6657 non-null   object
 7   City_code       6657 non-null   object
 8   Male            6657 non-null   object
 9   Time_Period     6657 non-null   object
 10  Weekday         6657 non-null   object
 11  Month           6657 non-null   object
 12  Year            6657 non-null   int64
 13  Clicked         6657 non-null   int64
dtypes: float64(3), int64(4), object(7)
memory usage: 728.2+ KB
```

```
Advertise_data.describe()
```

	VistID	Time_Spent	Age	Avg_Income	Internet_Usage	Year	Clicked
count	6.657000e+03	6657.000000	6657.000000	6657.000000	6657.000000	6657.0	6657.000000
mean	5.542115e+06	66.849548	37.258825	55930.486743	184.947684	2020.0	0.456362
std	2.596284e+06	15.509672	10.995458	13110.339257	43.189896	0.0	0.498129
min	1.000187e+06	32.600000	19.000000	13996.500000	104.780000	2020.0	0.000000
25%	3.307428e+06	55.200000	28.000000	48454.000000	145.730000	2020.0	0.000000
50%	5.523907e+06	70.660000	36.000000	58183.000000	193.580000	2020.0	0.000000
75%	7.823942e+06	79.570000	46.000000	65957.000000	222.260000	2020.0	1.000000
max	9.999708e+06	91.430000	61.000000	79484.800000	269.960000	2020.0	1.000000

```
Advertise_data.isnull().sum()
```

```
VistID          0
Time_Spent      0
Age             0
Avg_Income      0
Internet_Usage  0
Ad_Topic        0
Country_Name    0
City_code       0
Male            0
Time_Period     0
Weekday         0
Month           0
Year            0
Clicked         0
dtype: int64
```

```
selectedPredictor= ['Time_Spent','Age','Avg_Income','Internet_Usage','Ad_Topic','City_code', 'Male', 'Time_Period']
FinalData = Advertise_data[selectedPredictor]
FinalData.head()
```

	Time_Spent	Age	Avg_Income	Internet_Usage	Ad_Topic	City_code	Male	Time_Period
0	87.97	43	55901.12	185.46	product_11	City_5	No	Mid-Night
1	51.63	50	39132.00	176.73	product_8	City_1	No	Evening
2	82.37	38	57032.36	210.60	product_6	City_2	No	Morning
3	62.06	45	48868.00	190.05	product_19	City_3	Yes	Morning
4	77.66	31	61608.23	204.86	product_11	City_2	No	Noon

```
FinalData['Male'].replace({'Yes':1,'No':0},inplace= True)
FinalData.head()
```

	Time_Spent	Age	Avg_Income	Internet_Usage	Ad_Topic	City_code	Male	Time_Period
0	87.97	43	55901.12	185.46	product_11	City_5	0	Mid-Night
1	51.63	50	39132.00	176.73	product_8	City_1	0	Evening
2	82.37	38	57032.36	210.60	product_6	City_2	0	Morning
3	62.06	45	48868.00	190.05	product_19	City_3	1	Morning
4	77.66	31	61608.23	204.86	product_11	City_2	0	Noon

```
Final_numData= pd.get_dummies(FinalData)
Final_numData['Clicked']= Advertise_data['Clicked']
Final_numData.head()
```

	Time_Spent	Age	Avg_Income	Internet_Usage	Male	Ad_Topic_product_1	Ad_Topic_product_10	Ad_Topic_product_11	Ad_Topic_product_12	Ad_Topic_product_13	...
0	87.97	43	55901.12	185.46	0	0	0	1	0	0	...
1	51.63	50	39132.00	176.73	0	0	0	0	0	0	...
2	82.37	38	57032.36	210.60	0	0	0	0	0	0	...
3	62.06	45	48868.00	190.05	1	0	0	0	0	0	...
4	77.66	31	61608.23	204.86	0	0	0	1	0	0	...

5 rows × 51 columns

```
Final_numData= pd.get_dummies(FinalData)
Final_numData['Clicked']= Advertise_data['Clicked']
Final_numData.head()
Final_numData.columns
```

```
Index(['Time_Spent', 'Age', 'Avg_Income', 'Internet_Usage', 'Male',
       'Ad_Topic_product_1', 'Ad_Topic_product_10', 'Ad_Topic_product_11',
       'Ad_Topic_product_12', 'Ad_Topic_product_13', 'Ad_Topic_product_14',
       'Ad_Topic_product_15', 'Ad_Topic_product_16', 'Ad_Topic_product_17',
       'Ad_Topic_product_18', 'Ad_Topic_product_19', 'Ad_Topic_product_2',
       'Ad_Topic_product_20', 'Ad_Topic_product_21', 'Ad_Topic_product_22',
       'Ad_Topic_product_23', 'Ad_Topic_product_24', 'Ad_Topic_product_25',
       'Ad_Topic_product_26', 'Ad_Topic_product_27', 'Ad_Topic_product_28',
       'Ad_Topic_product_29', 'Ad_Topic_product_3', 'Ad_Topic_product_30',
       'Ad_Topic_product_4', 'Ad_Topic_product_5', 'Ad_Topic_product_6',
       'Ad_Topic_product_7', 'Ad_Topic_product_8', 'Ad_Topic_product_9',
       'City_code_City_1', 'City_code_City_2', 'City_code_City_3',
       'City_code_City_4', 'City_code_City_5', 'City_code_City_6',
       'City_code_City_7', 'City_code_City_8', 'City_code_City_9',
       'Time_Period_Early-Morning', 'Time_Period_Evening',
       'Time_Period_Mid-Night', 'Time_Period_Morning', 'Time_Period_Night',
       'Time_Period_Noon', 'Clicked'],
      dtype='object')
```

```
TargetVariable = 'Clicked'
Predictors = ['Time_Spent', 'Age', 'Avg_Income', 'Internet_Usage', 'Male',
       'Ad_Topic_product_1', 'Ad_Topic_product_10', 'Ad_Topic_product_11',
       'Ad_Topic_product_12', 'Ad_Topic_product_13', 'Ad_Topic_product_14',
       'Ad_Topic_product_15', 'Ad_Topic_product_16', 'Ad_Topic_product_17',
       'Ad_Topic_product_18', 'Ad_Topic_product_19', 'Ad_Topic_product_2',
       'Ad_Topic_product_20', 'Ad_Topic_product_21', 'Ad_Topic_product_22',
       'Ad_Topic_product_23', 'Ad_Topic_product_24', 'Ad_Topic_product_25',
       'Ad_Topic_product_26', 'Ad_Topic_product_27', 'Ad_Topic_product_28',
       'Ad_Topic_product_29', 'Ad_Topic_product_3', 'Ad_Topic_product_30',
       'Ad_Topic_product_4', 'Ad_Topic_product_5', 'Ad_Topic_product_6',
       'Ad_Topic_product_7', 'Ad_Topic_product_8', 'Ad_Topic_product_9',
       'City_code_City_1', 'City_code_City_2', 'City_code_City_3',
       'City_code_City_4', 'City_code_City_5', 'City_code_City_6',
       'City_code_City_7', 'City_code_City_8', 'City_code_City_9',
       'Time_Period_Early-Morning', 'Time_Period_Evening',
       'Time_Period_Mid-Night', 'Time_Period_Morning', 'Time_Period_Night',
       'Time_Period_Noon']

X= Final_numData[Predictors].values
y= Final_numData[TargetVariable].values
```

```
[ ] from sklearn.model_selection import train_test_split
    X_train,X_test,y_train,y_test = train_test_split(X,y , test_size = .2 , random_state = 42)
```

```
[ ] print(X_train.shape)
    print(X_test.shape)
    print(y_train.shape)
    print(y_test.shape)
```

```
(5325, 50)
(1332, 50)
(5325,)
(1332,)
```

```
[ ] from sklearn import tree
    model= tree.DecisionTreeClassifier()
    model.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[ ] pred_dt =model.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
tab1 = confusion_matrix(pred_dt , y_test)
tab1
```

```
array([[654,  74],
       [ 62, 542]])
```

```
[ ] tab1.diagonal().sum() / tab1.sum() * 100
```

```
89.7897897897898
```

```
[ ] model.feature_importances_
```

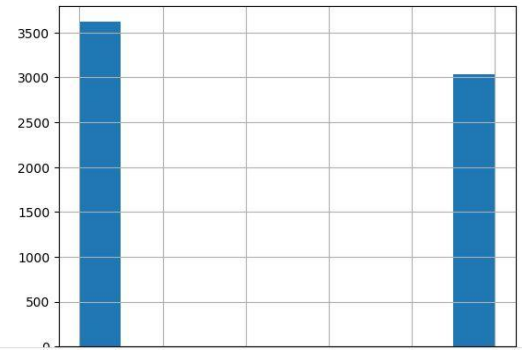
```
array([1.71564442e-01, 3.77251037e-02, 5.60664468e-02, 6.57581503e-01,
       1.57781907e-03, 2.43606496e-03, 3.56635714e-03, 1.75268139e-03,
       1.24472500e-03, 2.19020849e-03, 1.97250147e-04, 3.22871607e-04,
       1.49806441e-03, 3.53016681e-04, 1.25405350e-03, 1.51803477e-04,
       1.77049917e-04, 4.55182026e-04, 3.62445862e-04, 1.04867625e-03,
       7.90100379e-04, 1.13789175e-03, 1.71641796e-03, 1.28463160e-03,
       2.23867205e-03, 3.52669509e-03, 6.72899790e-04, 9.92615573e-04,
       1.25963711e-03, 2.12247843e-04, 1.38088007e-03, 4.85009148e-04,
       2.17907526e-03, 1.45254261e-03, 1.20377672e-03, 4.60600641e-03,
       2.83531073e-03, 1.83169492e-03, 9.94482533e-04, 4.95959566e-03,
       2.17992919e-03, 1.14863911e-04, 3.32578540e-04, 0.00000000e+00,
       5.23288641e-03, 5.22874212e-03, 2.93869479e-04, 1.94789010e-03,
       3.33145226e-03, 4.05183937e-03])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, pred_dt)
```

```
0.8978978978978979
```

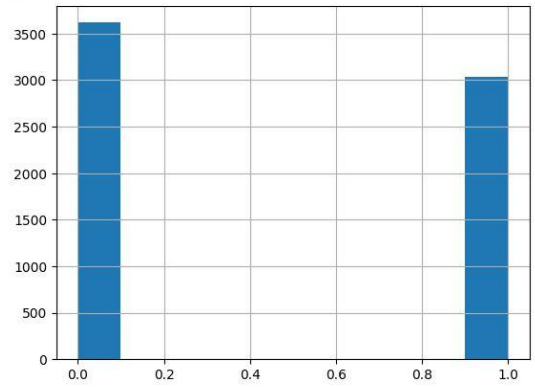
```
[ ] Advertise_data['Clicked'].hist()
```

```
<Axes: >
```

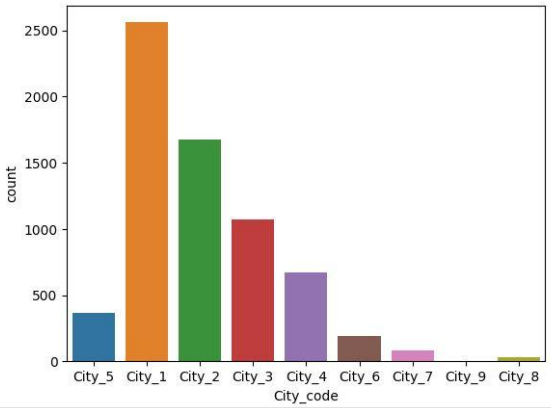



```
[ ] Advertise_data['Clicked'].hist()
```

```
<Axes: >
```

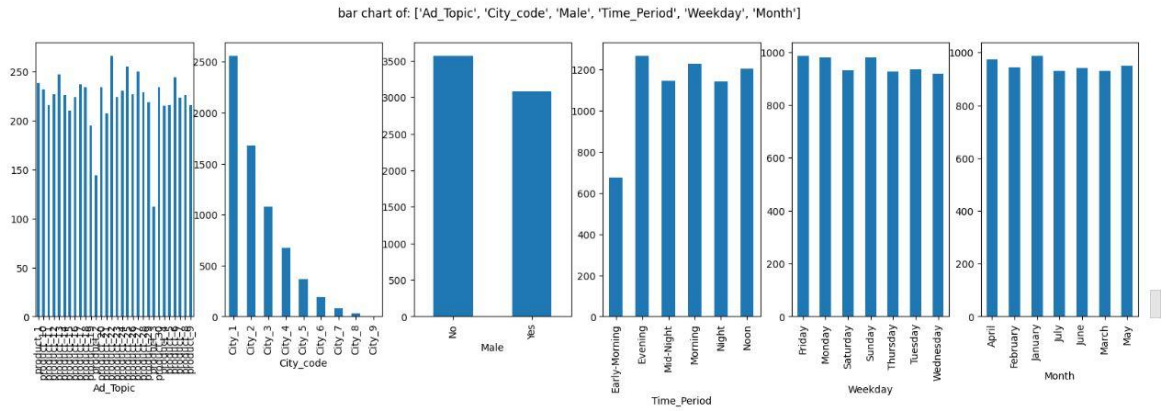



```
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x= 'City_code',data= Advertise_data)
plt.show()
```



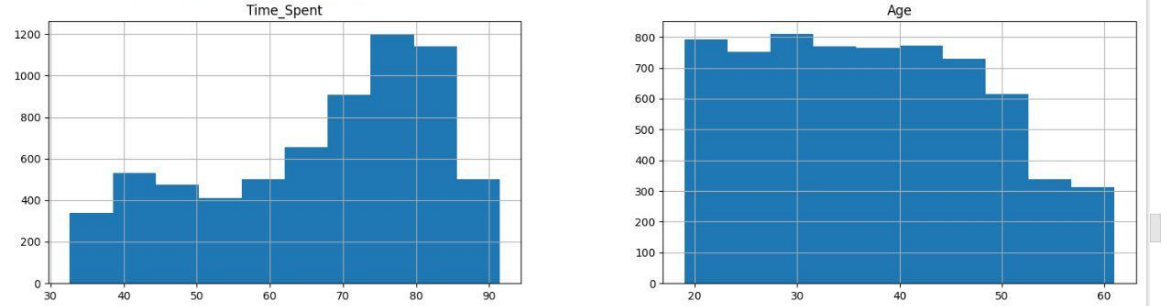

```
def plotbarchart(inpData, Colstoplot):
    %matplotlib inline

    import matplotlib.pyplot as plt


    fig, subplot = plt.subplots(nrows = 1, ncols= len(Colstoplot), figsize = (20,5))
    # OR fig, subplot = plt.subplots(nrows= len(Colstoplot),ncols=1, figsize = (5,15))
    fig.suptitle("bar chart of: " + str(Colstoplot))

    for colnum, plotnum in zip (Colstoplot, range(len(Colstoplot))):
        inpData.groupby(colnum).size().plot(kind='bar',ax= subplot[plotnum])
plotbarchart(inpData=Advertise_data , Colstoplot= ["Ad_Topic","City_code", "Male","Time_Period", "Weekday","Month"])
```

[ ]

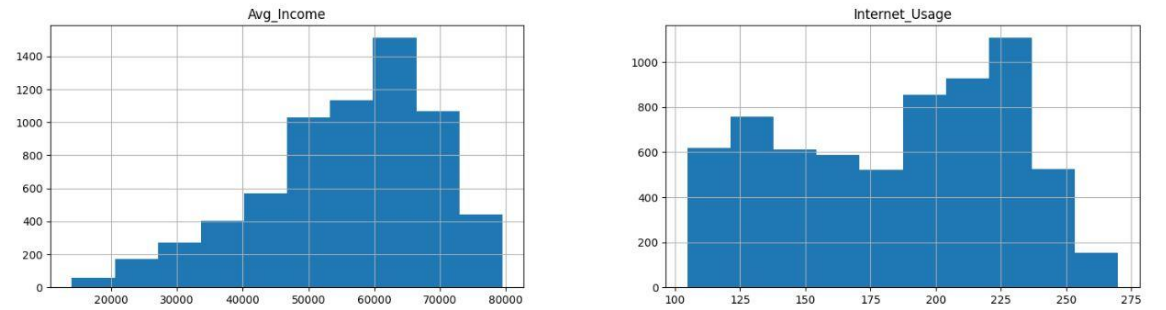



```
Advertise_data.hist(['Time_Spent','Age','Avg_Income','Internet_Usage','Ad_Topic'],figsize=(18,10))
```

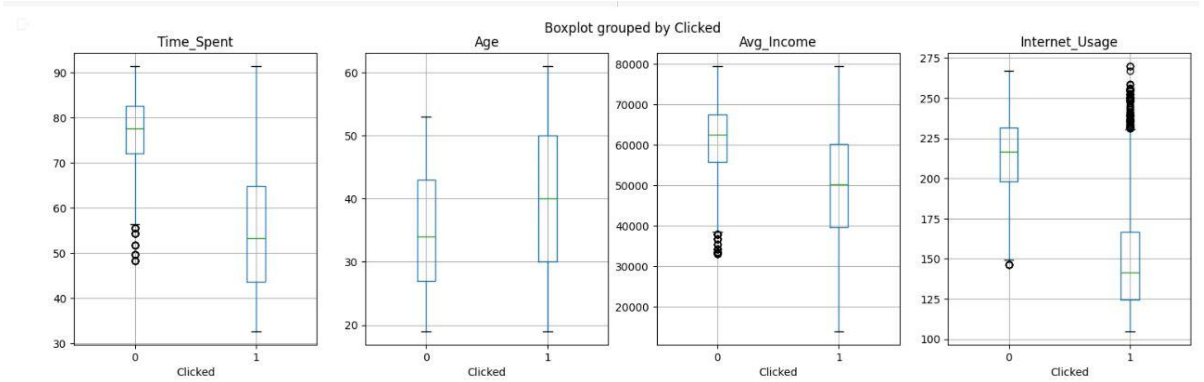
```
array([[<Axes: title={'center': 'Time_Spent'}>,
        <Axes: title={'center': 'Age'}>],
       [<Axes: title={'center': 'Avg_Income'}>,
        <Axes: title={'center': 'Internet_Usage'}>]], dtype=object)
```



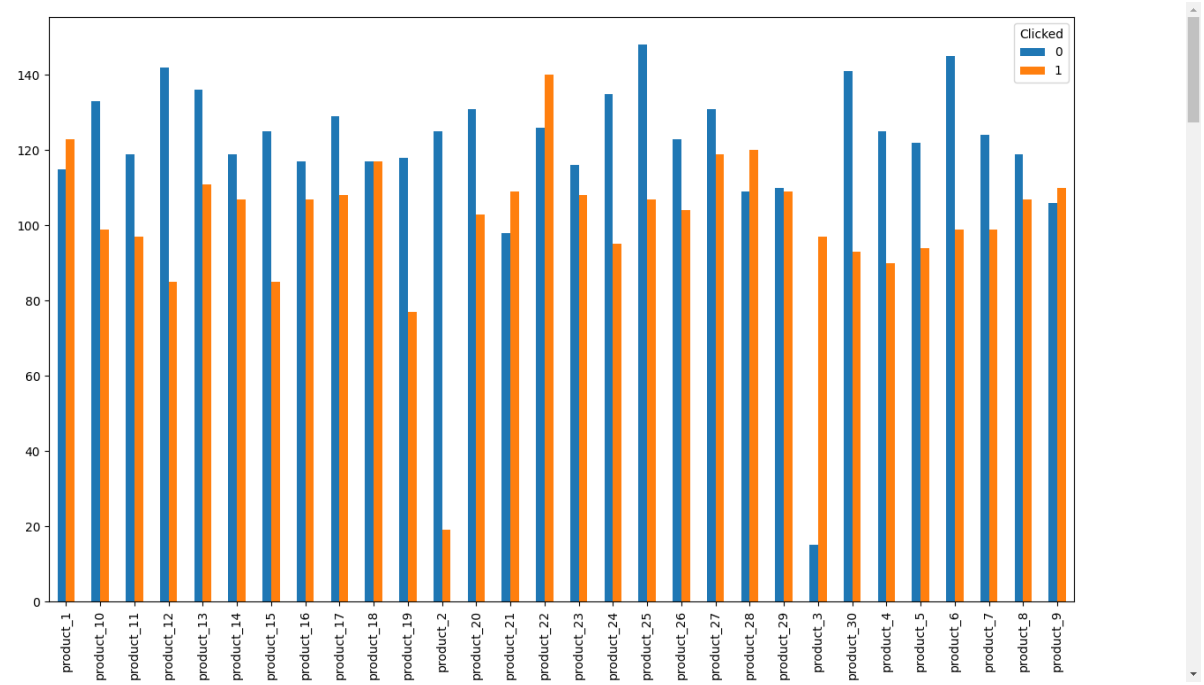
[ ]

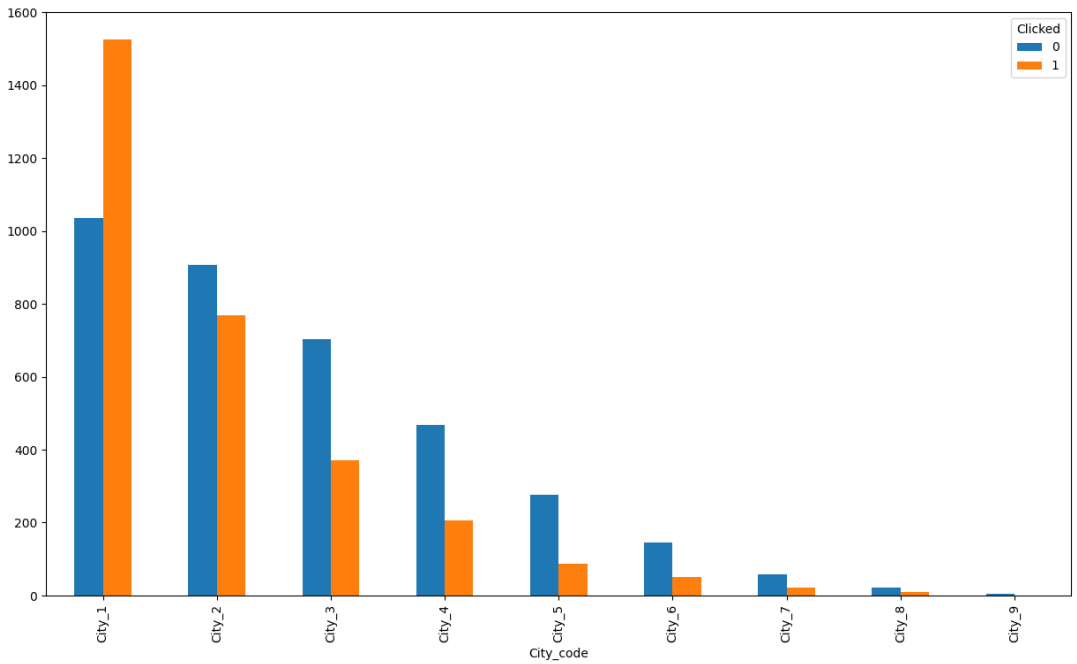


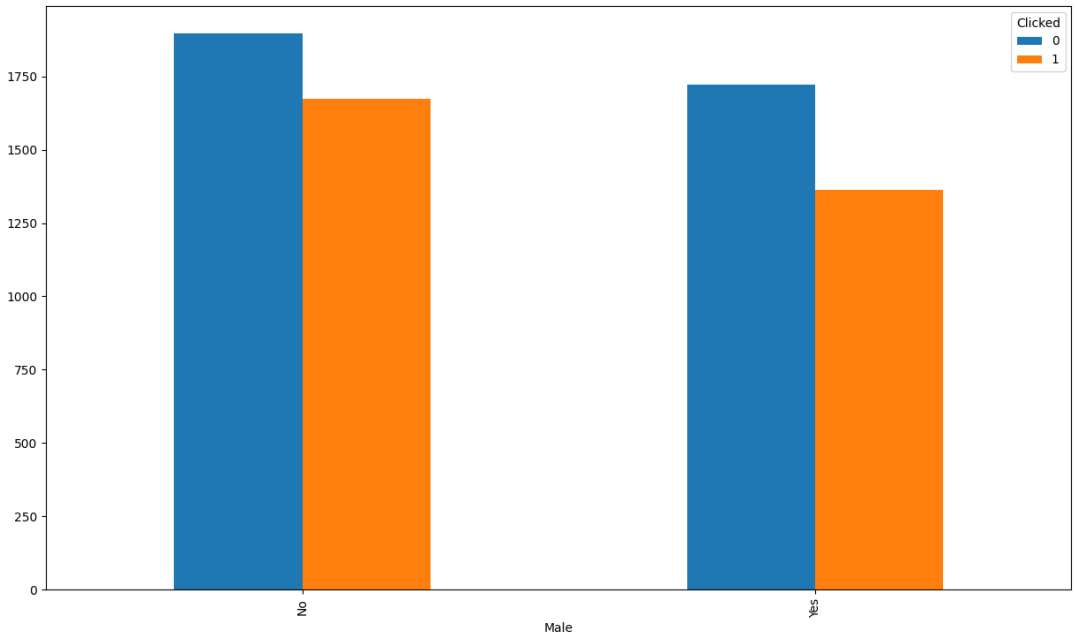

```
ContinuousColList= ['Time_Spent','Age','Avg_Income','Internet_Usage']

import matplotlib.pyplot as plt

fig, subplot = plt.subplots(nrows= 1, ncols= len (ContinuousColList), figsize= (18,5))

for i, plotnum in zip(ContinuousColList, range(len(ContinuousColList))):
    Advertise_data.boxplot(column = i, by= 'Clicked', figsize= (18,5), vert= True, ax=subplot[plotnum])
```

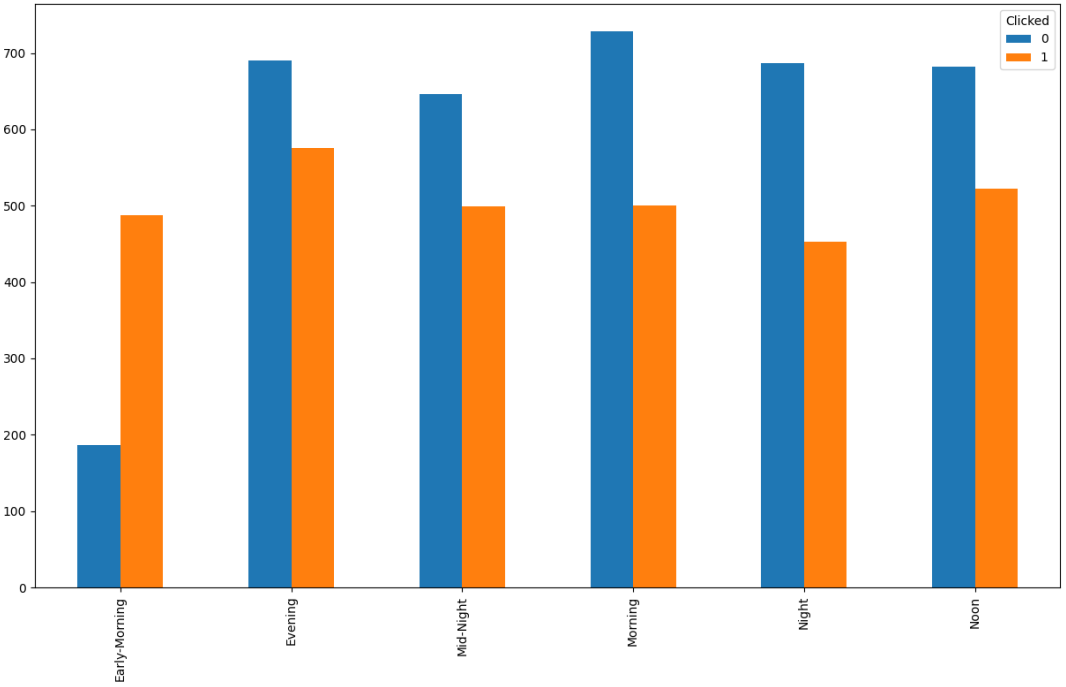


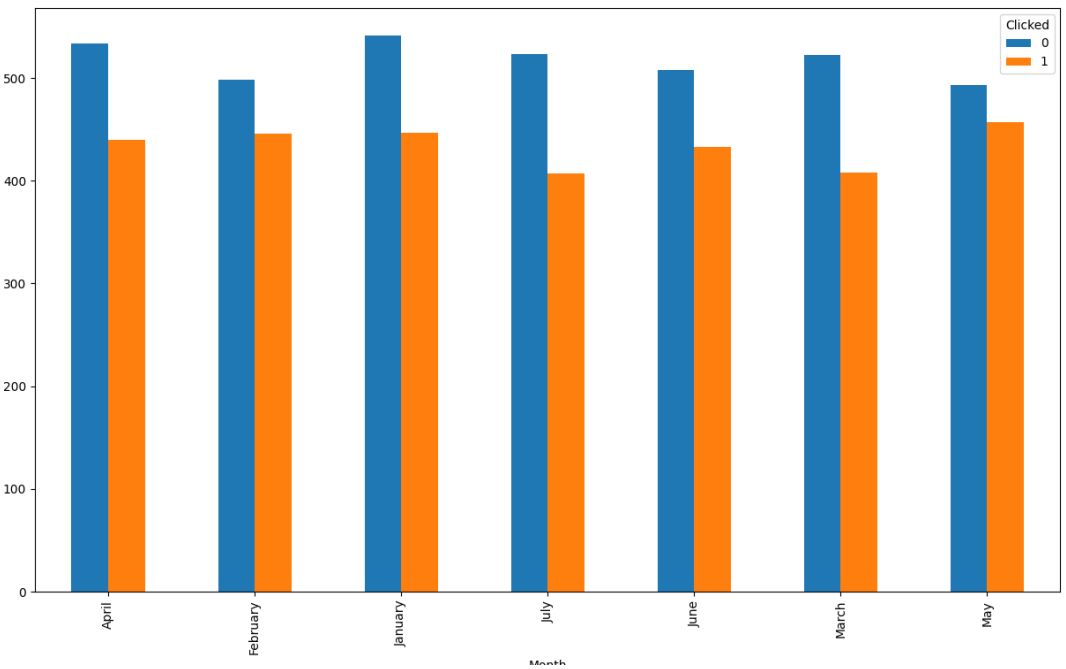

```
CategoricalList= ["Ad_Topic","City_code", "Male", "Time_Period", "Weekday","Month"]

import matplotlib.pyplot as plt

fig, subplot= plt.subplots(nrows= len(CategoricalList),figsize= (15,60))


for colnum, plotnum in zip( CategoricalList, range(len(CategoricalList))):
    crosstabResult= pd.crosstab(index= Advertise_data[colnum], columns= Advertise_data['Clicked'])
    crosstabResult.plot.bar(ax= subplot[plotnum])
```




Clicked
0
1
1600
1400
1200
1000
800
600
400
200
0
City_1
City_2
City_3
City_4
City_5
City_6
City_7
City_8
City_9
City_code


Clicked
0
1
1750
1500
1250
1000
750
500
250
0
No
Yes
Male


Clicked
0
1
0
100
200
300
400
500
600
700
Early-Morning
Evening
Mid-Night
Morning
Night
Noon


Clicked
0
1
0
100
200
300
400
500
April
February
January
July
June
March
May