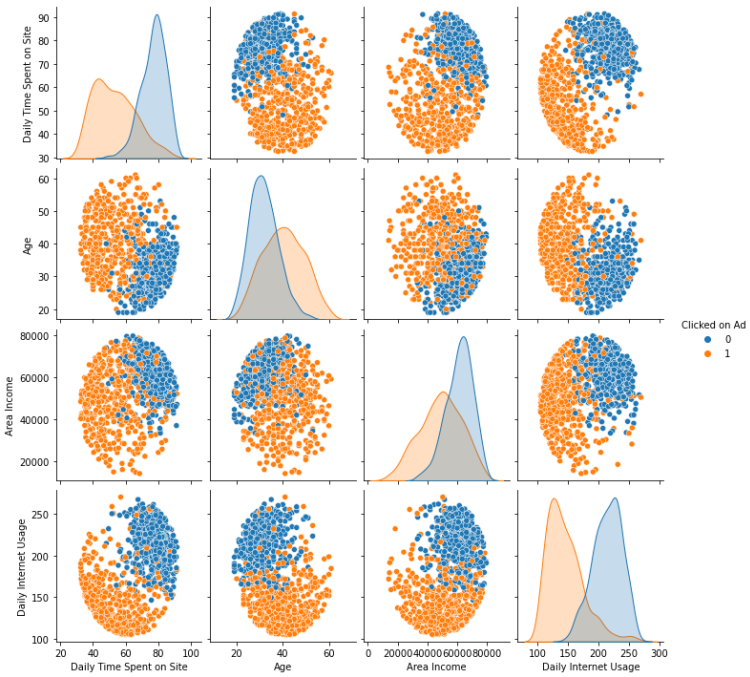


PROJECT REPORT

Machine learning models to predict if a user will click on an advertisement or not

ABSTRACT

This project aimed to generate different machine learning models to predict whether a user will click on an advertisement or not. The data consisted of ten variables with about a thousand entries for each variable. The data was cleaned, categorical features were transformed into numerical data and the correlation between the features and the target (click on ad) was calculated. Features with low correlation were excluded from the ML model calculations. The project corresponded to a binary classification and the ML models chosen were: Logistic Regression Model, Naive Bayes Classifier, Support Vector Machines, K-Nearest Neighbour, Decision Tree Classifier and Random Forest. For models requiring normalisation, RobustScaler normalisation was applied, as the feature distributions were asymmetric and had outliers. After training the models on 80% of the dataset and testing them on the remaining 20%, the best performing model was the logistic regression model, with an accuracy of 0.943 on the test set. The rest of the models also showed good predictive power, as their accuracy was higher than 0.92. Due to the high accuracy and interpretability of the result, the logistic regression model should be considered as the most suitable for predictions in this dataset.

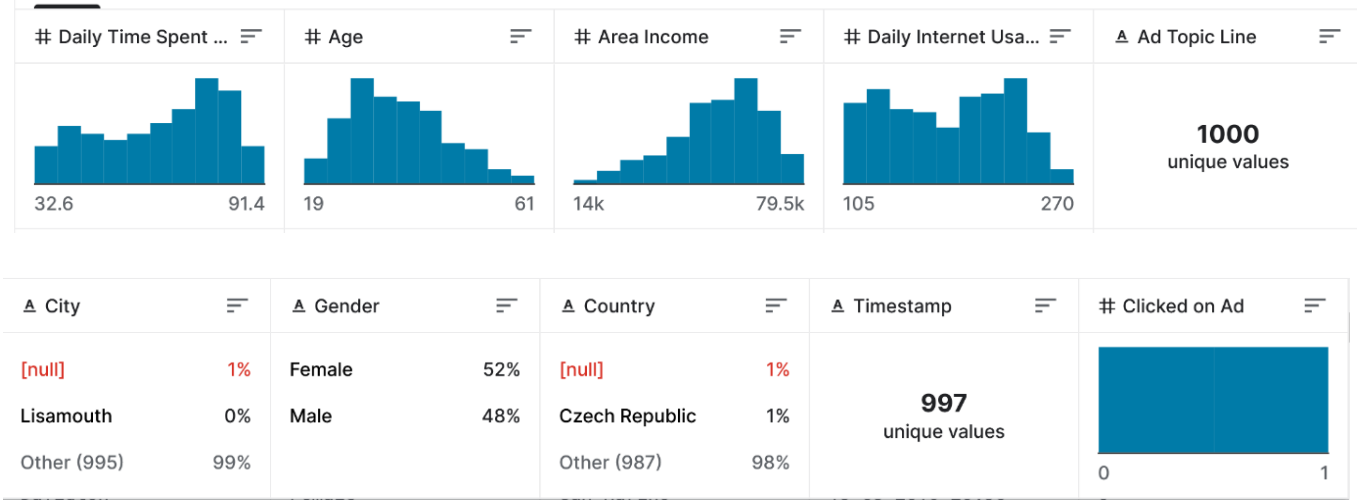
Priscila Boada

CONTENTS

DEFINITION OF THE PROBLEM

THE DATASET

'The data consists of 10 variables: 'Daily Time Spent on Site', 'Age', 'Area Income', 'Daily Internet Usage', 'Ad Topic Line', 'City', 'Male', 'Country', Timestamp' and 'Clicked on Ad'. The main variable is 'Clicked on Ad'. This variable can have two possible outcomes: 0 and 1 where 0 refers to the case where a user didn't click the advertisement, while 1 refers to the scenario where a user clicks the advertisement.'



Source: https://www.kaggle.com/datasets/hiimanshuagarwal/advertising-ef

```
df = pd.read_csv('advertising_ef.csv')
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Daily Time Spent on Site  1005 non-null   float64
 1   Age                       998 non-null    float64
 2   Area Income               998 non-null    float64
 3   Daily Internet Usage      1005 non-null   float64
 4   Ad Topic Line             1009 non-null   object
 5   City                      998 non-null    object
 6   Gender                    1009 non-null   object
 7   Country                   996 non-null    object
 8   Timestamp                 1009 non-null   object
 9   Clicked on Ad             1009 non-null   int64
dtypes: float64(4), int64(1), object(5)
memory usage: 79.0+ KB
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Gender	Country	Timestamp	Clicked on Ad
0	68.95	35.0	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	Female	Tunisia	27-03-2016 00:53	0
1	NaN	31.0	68441.85	193.77	Monitored national standardization	West Jodi	Male	Nauru	04-04-2016 01:39	0
2	69.47	26.0	59785.94	236.50	Organic bottom-line service-desk	Davidton	Female	San Marino	13-03-2016 20:35	0
3	74.15	29.0	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	Male	Italy	10-01-2016 02:31	0

OBJECTIVES

1. Perform some exploratory data analysis to see how 'Daily Time Spent on Site' in combination with 'Ad Topic Line' affects the user's decision to click on the advertisement.
2. Define the features the most relevant features to predict the value 'Clicked on Ad' variable.
3. Identify and create the machine-learning algorithms needed to predict the value 'Clicked on Ad' variable

DATA EXPLORATION AND PREPARATION

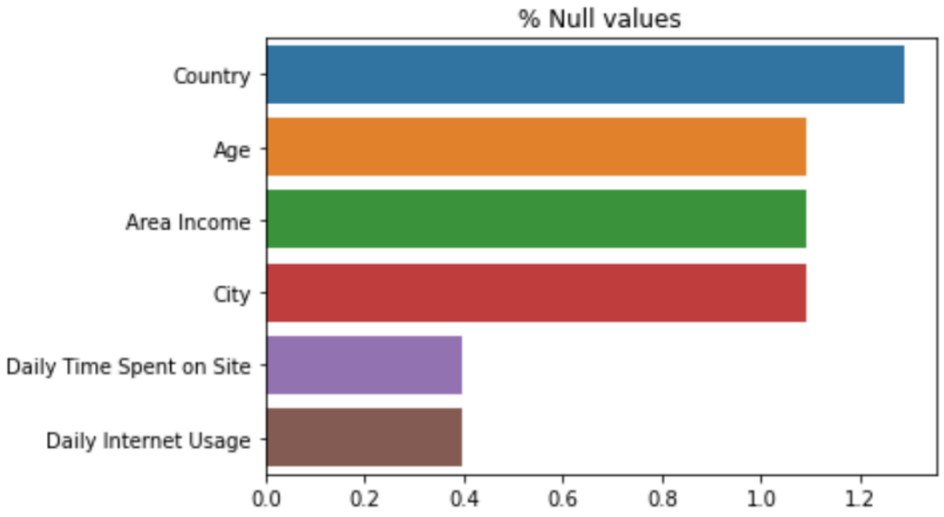
DATA CLEANING

FINDING MISSING DATA AND DEFINING ITS NATURE

The proportion of missing data was identified by using the .isnull method and the percentage of null values per feature was:

```
#Finding the % of null values per feature

df.isnull().mean().sort_values(ascending=False)*100
```

```
Country                     1.288404
Age                         1.090188
Area Income                 1.090188
City                        1.090188
Daily Time Spent on Site    0.396432
Daily Internet Usage        0.396432
Ad Topic Line               0.000000
Gender                      0.000000
Timestamp                   0.000000
Clicked on Ad               0.000000
dtype: float64
```



```
#Finding the total percentage of null values

df.isnull().values.mean()*100
```

```
0.535183349851338
```

Rule of decision: "Schafer (1999) asserted that a missing rate of 5% or less is inconsequential. Bennett (2001) maintained that statistical analysis is likely to be biased when more than 10% of data are missing."[i]

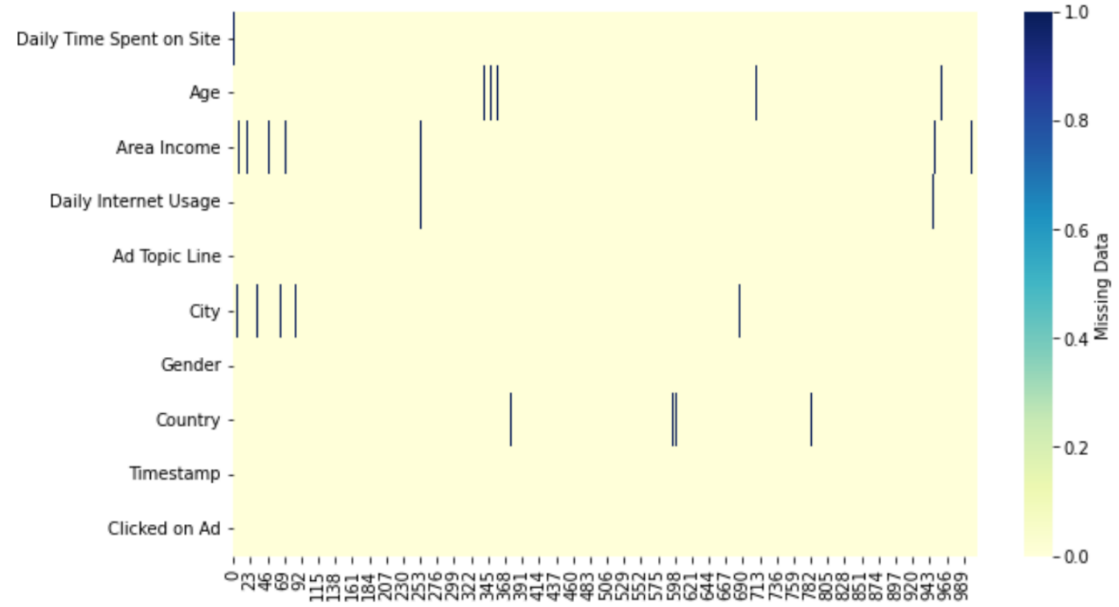
After applying the "rule of decision" above, it was concluded that the proportion of missing data was not biased since the percentage of the total null values is 0.53%

To define the mechanism of missing data Little's test was applied considering that the data could be classified as:

1. Missing Completely At Random (MCAR): No relationship between the fact that data is missing and either the observed or unobserved covariates
2. Missing At Random: The missingness is still random but can have some relationship with other variables in the data
3. Missing Not At Random: The Data missed is directly correlated with the value of the missing data.[ii]

A p. value of less than 0.05 is usually interpreted as being that the missing data is not MCAR (i.e., is either Missing At Random or non-ignorable).

```
plt.figure(figsize = (10,6))
sns.heatmap(df.isna().transpose(),
            cmap = 'YlGnBu',
            cbar_kws = {'label': 'Missing Data'})
plt.show()
```



Since the heatmap showed p.values>0.05 the data set was treated as MCAR

DEALING WITH MISSING DATA, DUPLICATES AND PRIVATE DATA

After proving that the data was not biased (%missing values = 0.54) and was Missing Completely at Ramdon (p.values>0.05), the delitation method was applied and the data set was modified as follows:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 956 entries, 0 to 1008
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Daily Time Spent on Site  956 non-null    float64
 1   Age                       956 non-null    float64
 2   Area Income               956 non-null    float64
 3   Daily Internet Usage      956 non-null    float64
 4   Ad Topic Line             956 non-null    object
 5   City                      956 non-null    object
 6   Gender                    956 non-null    object
 7   Country                   956 non-null    object
 8   Timestamp                 956 non-null    object
 9   Clicked on Ad             956 non-null    int64
dtypes: float64(4), int64(1), object(5)
memory usage: 82.2+ KB
```

To add more, there were no found duplicates nor unique values or private data that should be deleted, thus the data set was not further modified.

DATA PREPARATION

To find noisy data that may interfere with the creation of the ML models, an exploration of unique values was performed, finding that there were:

```
# Find unique values
df.nunique()
```

```
Daily Time Spent on Site    863
Age                          43
Area Income                 955
Daily Internet Usage        922
Ad Topic Line               955
City                        929
Gender                        2
Country                     237
Timestamp                   952
Clicked on Ad                 2
dtype: int64
```

As it is seen the dataset contained data that needed to be transformed to add meaning to the ML models. Thus, each feature was analyzed to find its value.

To determine the features that have numerical values and therefore are not categorical the describe() method was applied and the result was:

```
# Finding numerical values and their description
df.describe()
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Clicked on Ad
count	956.000000	956.000000	956.000000	956.000000	956.000000
mean	64.991381	36.024059	54994.663609	179.687887	0.505230
std	15.868862	8.804119	13306.592958	43.984389	0.500234
min	32.600000	19.000000	13996.500000	104.780000	0.000000
25%	51.360000	29.000000	46971.640000	138.647500	0.000000
50%	67.975000	35.000000	56985.410000	182.425000	1.000000
75%	78.585000	42.000000	65315.467500	218.182500	1.000000
max	91.430000	61.000000	79484.800000	269.960000	1.000000

From the information above, it was defined that the rest of the features are categorical values that needed a closer look.

DELETITION OF CATEGORICAL FEATURES

AD TOPIC

This column had 955 unique entries and since there was no further description of the category of the ad, there will not be considered for analysis. However, if it could be provided with an amplified description that allows categorizing the data, it would be highly recommended for analysis.

CITY

There are 929 unique entries which are more than 97% of this column. Thus, due to the high variation, this feature will not provide more information.

TRANSFORMATION OF CATEGORICAL FEATURES

GENDER

This feature was converted into nominal data following the next rule:

- If Gender is Female, turn into 1, else turn into 0

TIMESTAMP:

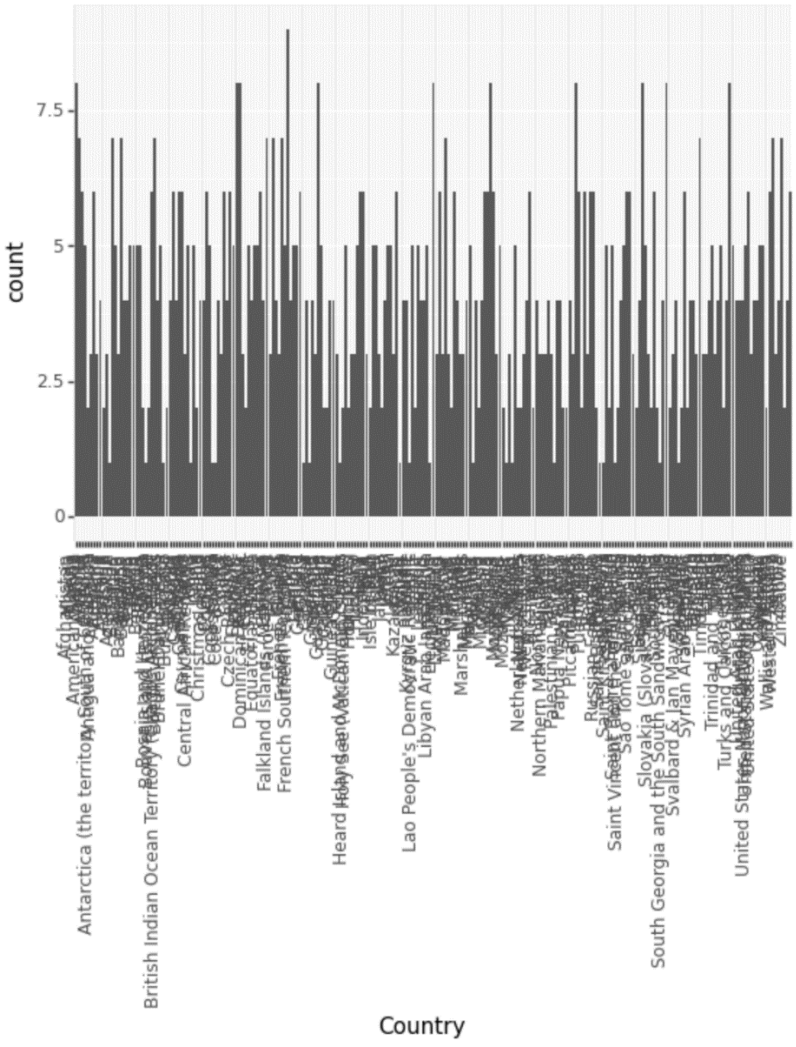
Since all the data belonged to the same year the approach taken was to transform into periods to spot any pattern. To do this, Pandas library was used and the to_datetime() method was applied. Then, the month, day of the week and hour were extracted and turned into new columns, resulting in:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Country	Clicked on Ad	Month	Day of the week	Hour 24
0	68.95	35.0	61833.90	256.09	1	Tunisia	0	3	6	0
2	69.47	26.0	59785.94	236.50	1	San Marino	0	3	6	21
3	74.15	29.0	54806.18	245.89	0	Italy	0	1	6	3
4	68.37	35.0	73889.99	225.58	1	Iceland	0	6	4	4
6	88.91	33.0	53852.85	208.36	1	Myanmar	0	1	3	21
...	...	...	...	...	...	...	...	...	...	...
1004	72.97	30.0	71384.57	208.58	0	Lebanon	1	2	3	22
1005	51.30	45.0	67782.17	134.42	0	Bosnia and Herzegovina	1	4	4	2
1006	51.63	51.0	42415.72	120.37	0	Mongolia	1	2	0	17
1007	55.55	19.0	41920.79	187.95	1	Guatemala	0	3	3	3
1008	45.01	26.0	29875.80	178.35	1	Brazil	1	6	4	22

956 rows × 10 columns

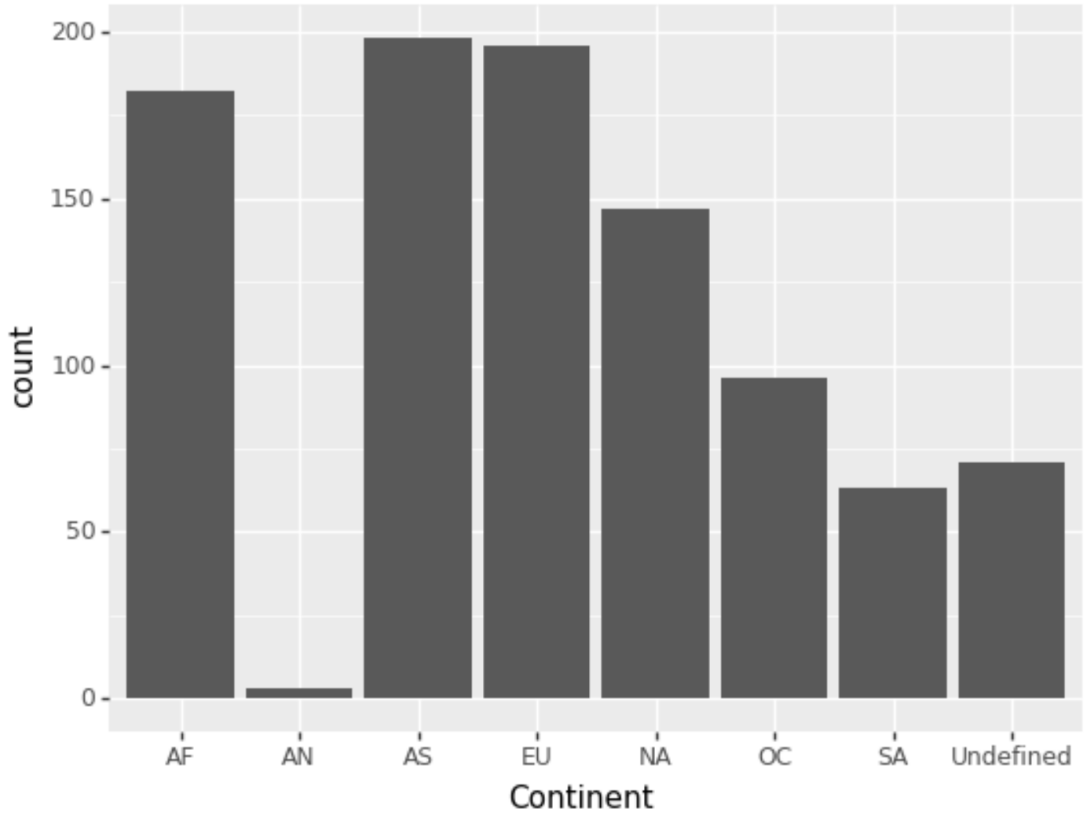
COUNTRY

This feature had 237 unique entries and showed highly disperse data as it is seen below:



To tackle the variability, the countries were grouped by continents. For this, the library, pycountry_convert, was used and the countries were assigned to their respective continent by using the ISO 3166-1 alpha-2 code. Since some countries did not fit with the code description, they had to be labelled as “Undefined”. The result was:

```
# Visualizing continent dispersion
ggplot(data = df_countries) + geom_bar(mapping = aes(x = 'Continent'))
```



To avoid the use of categorical variables, the Continents’ names were transformed into dummy variables and the feature Country was dropped since it does not provide further relevance.

```
df_countries = df_countries.drop(columns = ['Country'])
df_countries
```

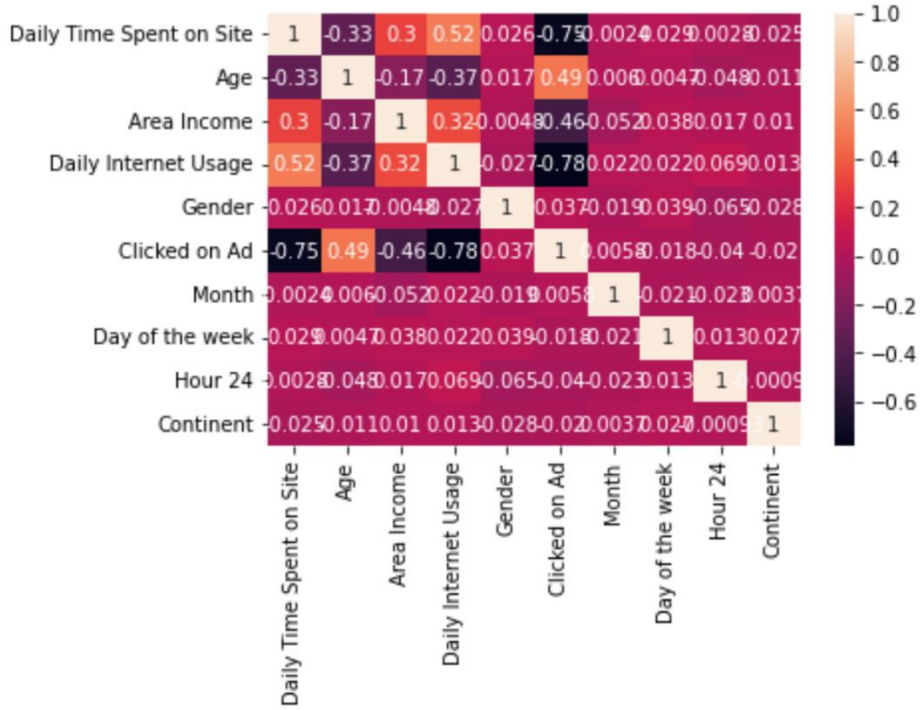
	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	Month	Day of the week	Hour 24	Continent
0	68.95	35.0	61833.90	256.09	1	0	3	6	0	1
2	69.47	26.0	59785.94	236.50	1	0	3	6	21	2
3	74.15	29.0	54806.18	245.89	0	0	1	6	3	2
4	68.37	35.0	73889.99	225.58	1	0	6	4	4	2
6	88.91	33.0	53852.85	208.36	1	0	1	3	21	3
...	...	...	...	...	...	...	...	...	...	...
1004	72.97	30.0	71384.57	208.58	0	1	2	3	22	3
1005	51.30	45.0	67782.17	134.42	0	1	4	4	2	2
1006	51.63	51.0	42415.72	120.37	0	1	2	0	17	3
1007	55.55	19.0	41920.79	187.95	1	0	3	3	3	4
1008	45.01	26.0	29875.80	178.35	1	1	6	4	22	7

956 rows × 10 columns

CORRELATION BETWEEN FEATURES

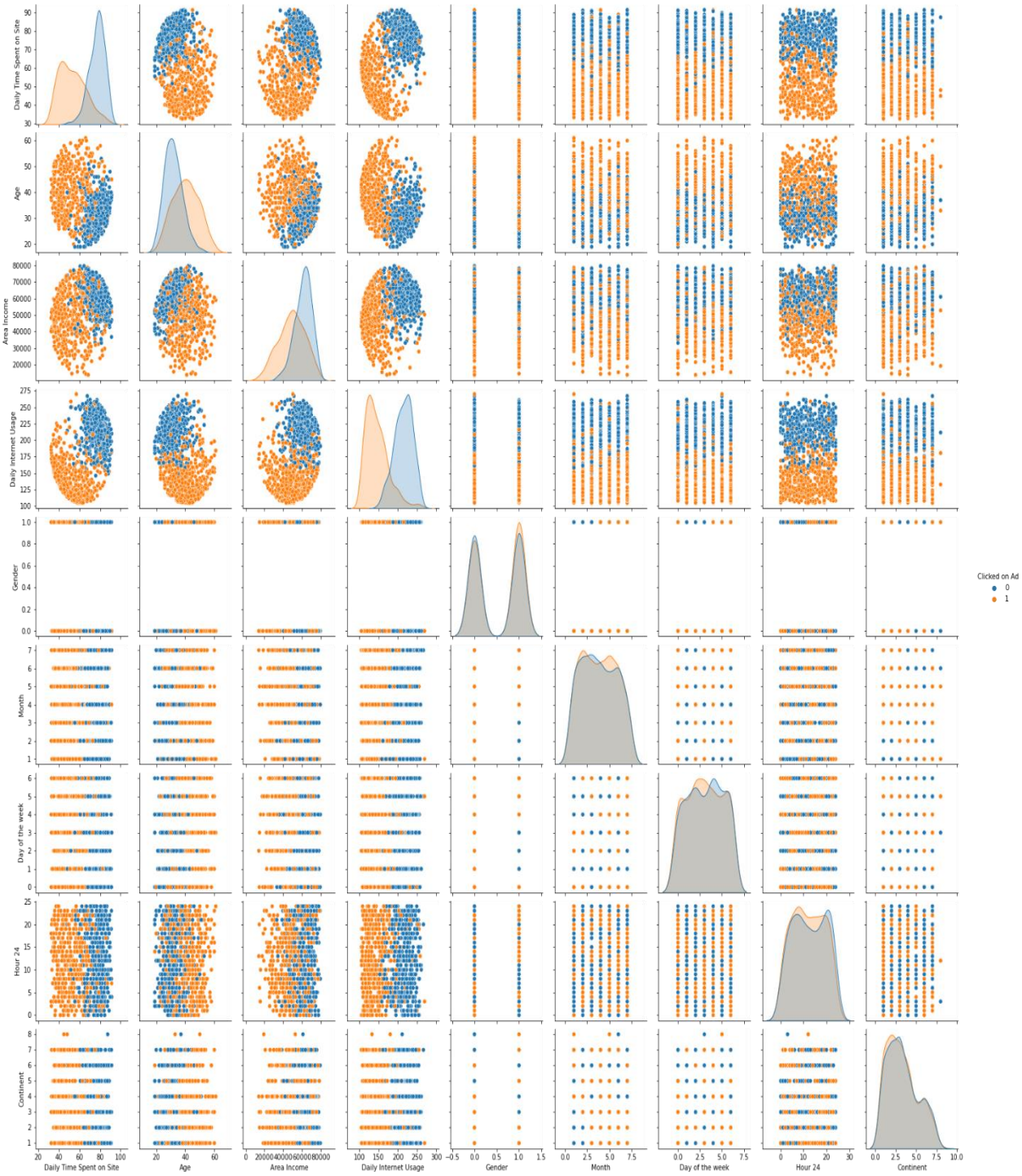
The Pearson correlation matrix was used to identify the correlations between the independent feature (Clicked on Ad) and the dependent features.

For this project, any correlation bigger than |±0.3| was considered for further investigation[iii]



	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	Month	Day of the week	Hour 24	Continent
Daily Time Spent on Site	1.000000	-0.330955	0.303340	0.515004	0.025826	-0.745277	-0.002430	0.028594	0.002808	-0.024945
Age	-0.330955	1.000000	-0.169398	-0.365944	0.017125	0.485594	0.005973	0.004723	-0.048461	-0.011458
Area Income	0.303340	-0.169398	1.000000	0.324678	-0.004775	-0.463475	-0.052185	0.037511	0.017111	0.010360
Daily Internet Usage	0.515004	-0.365944	0.324678	1.000000	-0.026930	-0.783238	0.021516	0.021525	0.068942	0.012746
Gender	0.025826	0.017125	-0.004775	-0.026930	1.000000	0.037199	-0.018965	0.038717	-0.064652	-0.028350
Clicked on Ad	-0.745277	0.485594	-0.463475	-0.783238	0.037199	1.000000	0.005823	-0.017598	-0.039524	-0.019940
Month	-0.002430	0.005973	-0.052185	0.021516	-0.018965	0.005823	1.000000	-0.021390	-0.022771	0.003724
Day of the week	0.028594	0.004723	0.037511	0.021525	0.038717	-0.017598	-0.021390	1.000000	0.013158	0.026869
Hour 24	0.002808	-0.048461	0.017111	0.068942	-0.064652	-0.039524	-0.022771	0.013158	1.000000	-0.000931
Continent	-0.024945	-0.011458	0.010360	0.012746	-0.028350	-0.019940	0.003724	0.026869	-0.000931	1.000000

Additionally, the different features and their distributions were plotted to find any patterns and correlations.



The features below had a Pearson Correlation smaller than |±0.3|, thus, they were not considered in the ML models.

- Gender
- Month
- Day of the week
- Hour 24
- Continent

Then the features to be modelled are:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Clicked on Ad
0	68.95	35.0	61833.90	256.09	0
2	69.47	26.0	59785.94	236.50	0
3	74.15	29.0	54806.18	245.89	0
4	68.37	35.0	73889.99	225.58	0
6	88.91	33.0	53852.85	208.36	0
...	...	...	...	...	...
1004	72.97	30.0	71384.57	208.58	1
1005	51.30	45.0	67782.17	134.42	1
1006	51.63	51.0	42415.72	120.37	1
1007	55.55	19.0	41920.79	187.95	0
1008	45.01	26.0	29875.80	178.35	1

956 rows × 5 columns

BALANCING DATA

To check if the target feature, Clicked on Ad, was balanced, the method value_counts() was applied. Then, the percentage of data labelled as 1 value was computed. Since 50.5% of the data had a value of 1 in Clicked on Ad, the dataset did not require further processing.

SOLUTIONS TO THE PROBLEM

Given the nature of the data, the ML models applied were classification models and the variable to be predicted (target) was 'Clicked on Add'.

For this binary classification, it was considered that a value of 1 represents clicked on Ad and 0 the opposite.

SELECTION OF MACHINE LEARNING MODELS

Considering the computing implications of each algorithm, the models were divided into:

- ML models not affected by standardization: Naive Bayes, Random Forest, Decision trees
- ML models that perform better with standardization: Support Vector Machines, K-nearest Neighbours, Logistic regression

To generate a pipeline to feed the data into the machine Learning algorithms, the following steps were taken:

1. **Definition of the explanatory and target variables**
   The target, Clicked on Ad, was stored under the variable targets. This was done by slicing the data frame.

```
targets = df['Clicked on Ad']
targets
```

```
0      0
1      0
2      0
3      0
4      0
      ..
951    1
952    1
953    1
954    0
955    1
Name: Clicked on Ad, Length: 956, dtype: int64
```

The explanatory variables, Daily Time Spent on Site, Age, Area Income and Daily Internet Usage were stored under the variable unscaled_inputs.

```
unscaled_inputs=df.iloc[:,:-1]
unscaled_inputs
```

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage
0	68.95	35.0	61833.90	256.09
1	69.47	26.0	59785.94	236.50
2	74.15	29.0	54806.18	245.89
3	68.37	35.0	73889.99	225.58
4	88.91	33.0	53852.85	208.36
...	...	...	...	...
951	72.97	30.0	71384.57	208.58
952	51.30	45.0	67782.17	134.42
953	51.63	51.0	42415.72	120.37
954	55.55	19.0	41920.79	187.95
955	45.01	26.0	29875.80	178.35

956 rows × 4 columns

2. **Splitting of the dataset into training and target**

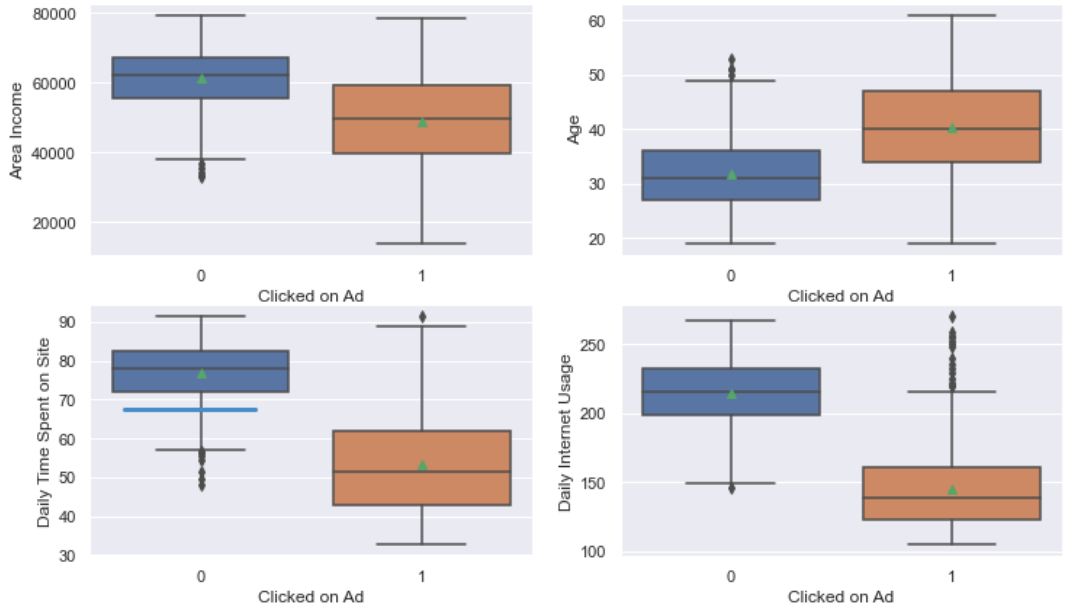
80% of the data was assigned for training while the remaining 20% was used for testing purposes. To ensure that the results are reproducible a random_state of 10 was fixed.

```
x_train, x_test, y_train, y_test = train_test_split(unscaled_inputs,targets, train_size = 0.8, random_state = 10)
print(x_train.shape,y_train.shape)
print(x_test.shape,y_test.shape)
```

```
(764, 4) (764,)
(192, 4) (192,)
```

3. **Standardization**

For the models that have a better performance when the data is scaled, the RobustScaler standardization was applied since the distribution of the features was skewed and outliers were identified. [iv]



To avoid data leakage, the RobustScaler standardization was performed on training and test data separately.

```
RS_x_train = RobustScaler ()
x_train_RS = RS_x_train.fit_transform(x_train)
print(x_train_RS.shape)

RS_x_test = RobustScaler ()
x_test_RS = RS_x_test.fit_transform(x_test)
print(x_train_RS.shape)
```

```
(764, 4)
(764, 4)
```

4. **Models training**

Each model was trained by using the sklearn library and its training accuracy was computed.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
```

5. **Models testing**

To check the best ML algorithm for this data set, the accuracy (ratio of correctly predicted instances divided by the total number of instances in the dataset multiplied) was computed and confusion matrixes were created to have a better visualization of the results.

ML MODELS NOT AFFECTED BY STANDARDIZATION

NAIVE BAYES CLASSIFIER

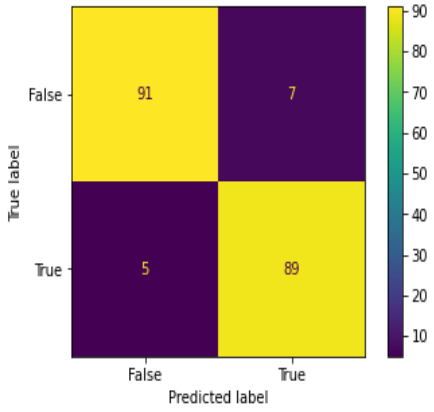
```
#Accuracy of the model with the test set
NB_accuracy_test = NBmodel.score(x_test,y_test)
NB_accuracy_test
```

```
0.9375
```

```
#Metrics to evaluate the model
y_predicted = NBmodel.predict(x_test)
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        98
           1       0.93      0.95      0.94        94

    accuracy                           0.94       192
   macro avg       0.94      0.94      0.94       192
weighted avg       0.94      0.94      0.94       192
```

```
#Confusion matrix to compare
confusion_matrix = metrics.confusion_matrix(y_test, y_predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```



Interpretation: The Naive Bayes Classifier model was able to classify the "Clicked on Ad" target with 93.75% accuracy on the test set. However, it should be noted that this algorithm takes all features as independent and also assumes a normal distribution. The dataset showed some correlations between features, as seen in the Pearson Correlation heatmap. In addition, the data had skewed distributions, so the condition of having a normal distribution was not fulfilled.

DECISION TREE CLASSIFIER (DTC)

The model was trained with different depths and it was seen that the model with the best performance had a depth of 4. Thus, that model was taken for further calculations.

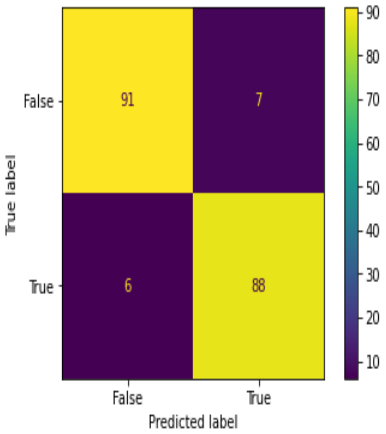
```
#train the model max_depth=4
DTmodel_deep4 = DecisionTreeClassifier(max_depth=4, random_state = 10)
DTmodel_deep4 = DTmodel_deep4.fit(x_train, y_train)
# Accuracy of the model with the training set
DT_accuracy_train_deep4 = DTmodel_deep4.score(x_train, y_train)
print(f'the training accuracy is:{DT_accuracy_train_deep4}')
DT_accuracy_test_deep4 = DTmodel_deep4.score(x_test,y_test)
print(f'the test accuracy is: {DT_accuracy_test_deep4}')
```

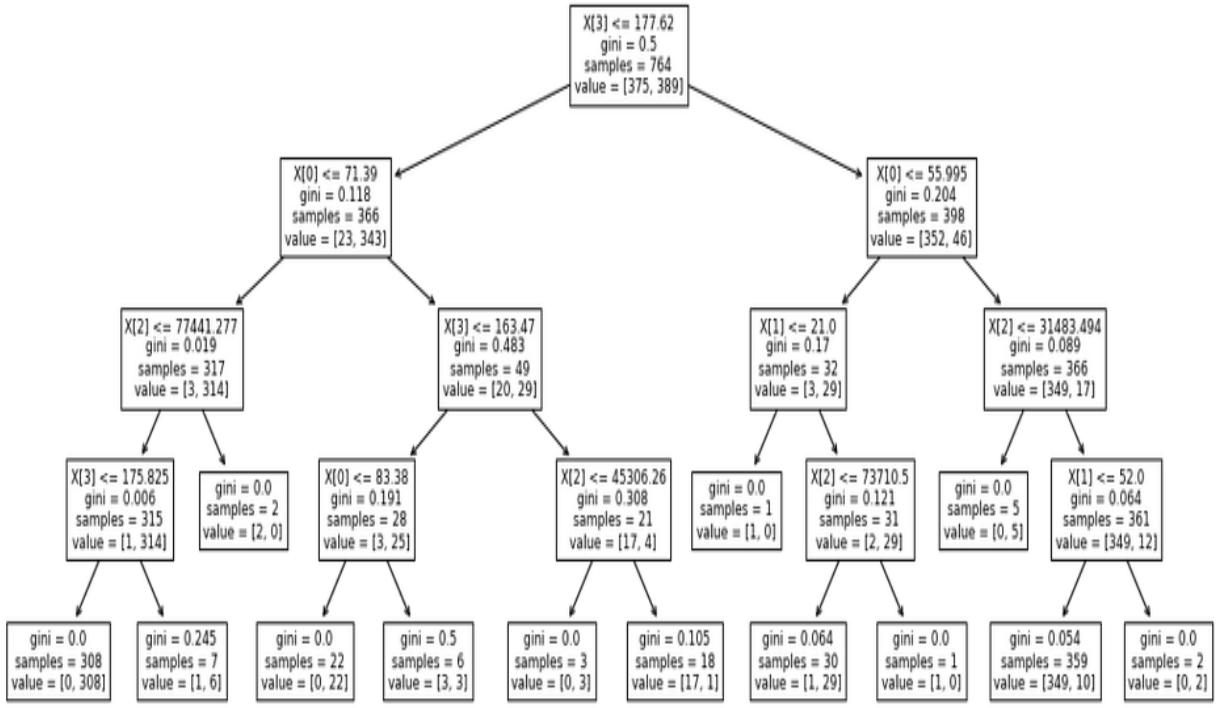
```
the training accuracy is:0.9790575916230366
the test accuracy is: 0.9322916666666666
```

```
#Metrics to evaluate the model
y_predicted =DTmodel_deep4.predict(x_test)
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.94      0.93      0.93        98
           1       0.93      0.94      0.93        94

    accuracy                           0.93       192
   macro avg       0.93      0.93      0.93       192
weighted avg       0.93      0.93      0.93       192
```

```
#Confusion matrix to compare
confusion_matrix = metrics.confusion_matrix(y_test, y_predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```





Interpretation: For this model, the feature with the highest weight was ' Daily Internet Usage', as it was located at node zero. Next, the second most relevant feature was 'Daily Time Spent on Site'. After this, the nodes of the tree were distributed along the different features obtaining a model that predicted the 'Clicked on Ad' target with 93.23% accuracy in the test set. [v]

RANDOM FOREST CLASSIFIER (RFC)

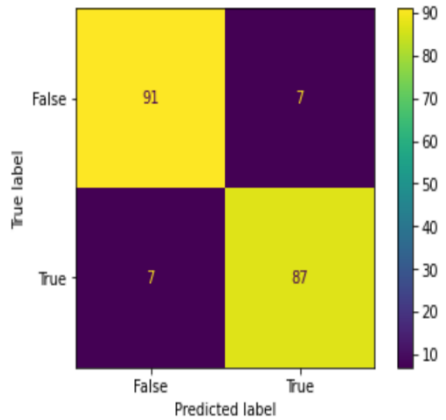
```
#Accuracy of the model with the test set
RF_accuracy_test = RFmodel.score(x_test,y_test)
RF_accuracy_test
```

```
0.9270833333333334
```

```
#Metrics to evaluate the model
y_predicted =RFmodel.predict(x_test)
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        98
           1       0.93      0.93      0.93        94

    accuracy                           0.93       192
   macro avg       0.93      0.93      0.93       192
weighted avg       0.93      0.93      0.93       192
```

```
#Confusion matrix to compare
confusion_matrix = metrics.confusion_matrix(y_test, y_predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```



Interpretation: The ** Random Forest Classifier ** classified the 'Clicked on Ad' target with 92.71% accuracy in the test set. Given that the RFC creates random decision trees to generate the model and therefore generates multiple layers that are more difficult to interpret, thus, the DTC might be a better option since has an easier interpretation and a higher accuracy level.[vi]

ML MODELS AFFECTED BY STANDARDIZATION

LOGISTIC REGRESSION MODEL

```
summary_table.index = summary_table.index + 1
#shifts up all the indices by 1
summary_table.loc[0] = ['Intercept',reg.intercept_[0]]
summary_table = summary_table.sort_index()
print('Logistic regression Coefficientes')
print(summary_table)
```

```
Logistic regression Coefficientes
                Feature_name  Coefficient
0                  Intercept     0.405756
1  Daily Time Spent on Site    -3.689405
2                        Age     1.788841
3                Area Income    -1.813051
4      Daily Internet Usage    -3.781988
```

The equation for this model is:

Logit(p) = 0.406 -3.782 * Daily Internet Usage - 3.689 * Daily Time Spent on Site - 1.813 * Area Income + 1.789 * Age

Test the model

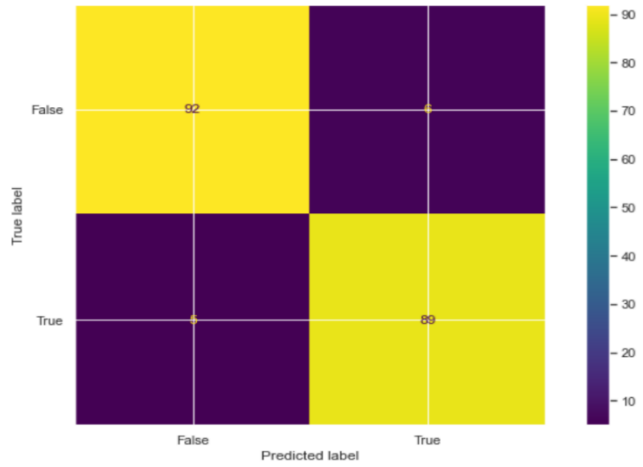
```
#Accuracy of the model with the test set
LRM_accuracy_test = reg.score(x_test_RS,y_test)
LRM_accuracy_test
```

```
[37]: 0.9427083333333334
```

```
#Metrics to evaluate the model
y_predicted = reg.predict(x_test_RS)
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.95      0.94      0.94        98
           1       0.94      0.95      0.94        94

    accuracy                           0.94       192
   macro avg       0.94      0.94      0.94       192
weighted avg       0.94      0.94      0.94       192
```

```
#Confusion matrix to compare
confusion_matrix = metrics.confusion_matrix(y_test, y_predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```



Interpretation: The **logistic regression model** was able to classify the "Clicked on Ad" target with 94.27% accuracy in the test set. In addition, this model provided access to coefficients that allowed determining which variables had a higher weight in the prediction of the model.

- A unit increase in Daily Internet Usage resulted in 3.782 decreases in logit (p) which translated into a 97% increase in the odds of clicking on the Ad.
- A unit increase in Daily Time Spent on Site resulted in 3.689 decreases in logit (p), which translated into a 97% increase in the odds of clicking on the Ad.
- A unit increase in Area Income resulted in 1.813 decreases in logit (p), which translated into an 83% increase in the odds of clicking on the Ad.
- A unit increase in Age resulted in 1.789 increases in logit (p), which translated into a 498% decrease in the odds of clicking on the Ad.

This model showed that the feature age has the highest impact on the computation of the models

K-NEAREST NEIGHBOUR

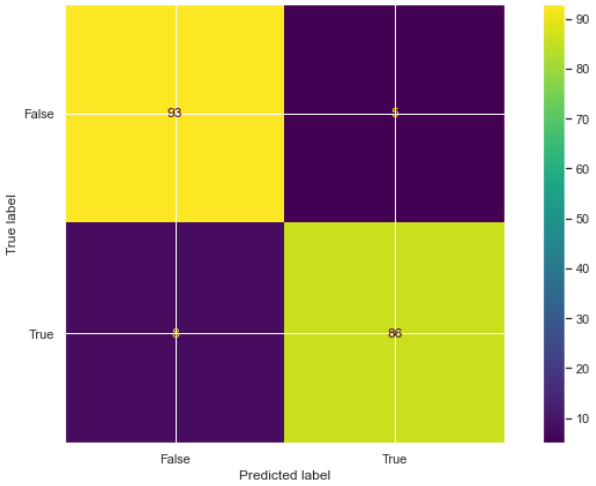
```
#Accuracy of the model with the test set
KNN_accuracy_test = KNNmodel.score(x_test_RS,y_test)
KNN_accuracy_test
```

```
: 0.9322916666666666
```

```
#Metrics to evaluate the model
y_predicted = KNNmodel.predict(x_test_RS)
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.93        98
           1       0.95      0.91      0.93        94

    accuracy                           0.93       192
   macro avg       0.93      0.93      0.93       192
weighted avg       0.93      0.93      0.93       192
```

Confusion matrix to compare



Interpretation: Since we are dealing with a binary classification, the number of neighbours could be set beforehand and this provided a model with a 93.2% of accuracy.

SUPPORT VECTOR MACHINES

```
# Accuracy of the model with the training set
SVM_accuracy_train = SVMmodel.score(x_train_RS, y_train)
SVM_accuracy_train
```

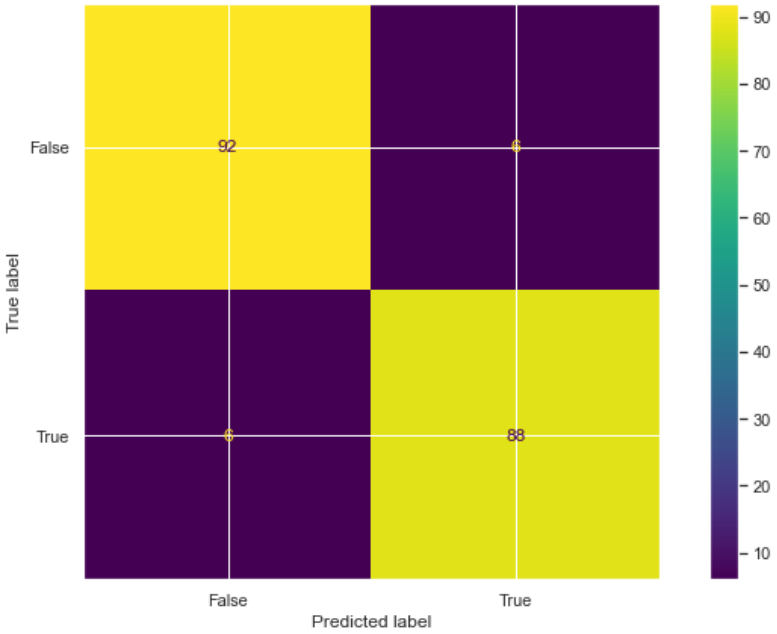
```
0.9764397905759162
```

```
#Accuracy of the model with the test set
SVM_accuracy_test = SVMmodel.score(x_test_RS,y_test)
SVM_accuracy_test
```

```
0.9375
```

```
#Metrics to evaluate the model
y_predicted = SVMmodel.predict(x_test_RS)
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        98
           1       0.94      0.94      0.94        94

    accuracy                           0.94       192
   macro avg       0.94      0.94      0.94       192
weighted avg       0.94      0.94      0.94       192
```



Interpretation: Since we are dealing with a binary classification, the number of vector machines could be set beforehand and this provided a model with a 93.8% of accuracy. Due to its nature, this model might offer some difficulty in interpretation. Thus, KNN model could be considered as a better fit

OUTCOME

The target 'Clicked on Ad' was able to be predicted using several ML models and all of them had high accuracy (above 93% in the test set).

Comparing the values of accuracy it is seen that the best ML model for this dataset is the Logistic regression model with a test accuracy of 94.3%. Since this model offers high interpretability and also high accuracy for this dataset, the Logistic regression model is recommended to be chosen to predict if a user will click on an advertisement or not.

	Model	training set	test set
0	Logistic regression model	0.969895	0.942708
1	Naive Bayes Classifier	0.969895	0.937500
2	Support Vector Machines	0.976440	0.937500
3	Decision Tree Classifier	0.979058	0.932292
4	Random Forest Classifier	0.996073	0.932292
5	K-Nearest Neighbour	0.977749	0.932292

LIMITATIONS OF THE INVESTIGATION

One of the aims of this project was to find out how the 'Ad Topic Line' affects the user's decision to click on the ad. However, this feature had a high variability (955 unique entries out of 956 entries) and no additional description to group topics into reasonable categories, so it had to be dismissed. Nonetheless, for future research, it would be worthwhile to have a deeper understanding of the ad's subject line in order to develop the categorisation.

REFERENCES

[i] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3701793/
[ii] https://soprasteriaanalytics.se/2020/01/23/an-easy-way-to-deal-with-missing-data-imputation-by-regression/
[iii] https://www.questionpro.com/blog/wp-content/uploads/2020/04/Pearson1-1.png
[iv] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html
[v] https://github.com/ValentinRicher/understanding-decision-trees/blob/master/understanding_decision_trees.ipynb
[vi] https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/