Predicting the user's decision to click on ads based on historical data

NICE
Software Solutions

Index

Problem Statement ---------- 3

Analysis ---------- 3

Model ---------- 3

Language ---------- 4

Importing Packages ---------- 4

Reading data from csv file ---------- 4

Defining Input and Output columns---------- 5

Importing library to split train and test sets---------- 6

Importing library to transform train and test sets ---------- 6

Import function to implement the model---------- 7

Importing function to create confusion matrix---------- 8

Importing calculate accuracy---------- 8

PROBLEM STATEMENT:

The business Owners/Marketing Managers needs to know if the user will click on ad or not based on the past experiences recorded by them. The analysis is to be done based on customer's Age, Salary and Gender.

The input parameters are already provided by the client. The correlation of the input columns with the output column is already studied and the irrelevant input columns are ruled out.

The important input columns which has an impact over the output that is "customer clicked on the ad" are Age, Salary and Gender.

ANALYSIS:

The analysis which is performed here is categorized as Classification Under supervised machine learning. The Supervised Machine learning or Classification method is used when the predicted out is defined or known for example True or False, Yes or No. The output values can be more than one. But are defined already.

Note:
While creating test and train dataset, it must be made sure that all the expected output values are present in the training dataset.

MODEL:

The model used here is KNN (K nearest neighbor) where K denotes the numbers of most matching values to be considered while prediction.

The model will predict if the new user with given Age, Salary and Gender will click on the ad or not.

LANGUAGE:

The language used to perform this analysis is Python 3.4.1. The IDE used for coding is Jupyter Notebook.

STEP 1:

The relevant packages are imported first to perform data import, data manipulation, numerical operations and transformation of data, and plotting graphs.

The packages imported are:

Numpy

Pandas

Matplotlib.pyplot

Importing Packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pt
```

STEP 2:

The next step is to read the data from csv file.

The csv file is stored at a location which is set as the working directory for this notebook.

Reading data from csv file

```
dataset= pd.read_csv("original.csv")
dataset
```

Also, the when the dataset is printed on the console, it displays the result as imported file.

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

The Clicked column denotes, whether the customer has clicked on the ad or not.

STEP 3:

The next step is to define the input and the output columns and store them in variables. The columns are stored as data frames.

The columns are to be stored separately as the function takes 4 parts as input which are Input training, Output training, Input test, Output test.

Defining Input and Output columns

```
X = dataset.iloc[:,[2,3]]
```

```
Y= dataset.iloc[:,4]
```

STEP 4:

In this step the training and test sets will be separated in order to train the model on the training set and test on the test set.

Here train_test_split function is imported from the library sklearn in which the folder model_selection contains that function.

The parameters of the function are given, and the data is distributed amongst X_train, X_test, Y_train and Y_test.

Here the training and test sets are separated in the ratio 75:25 which is specified as a function parameter.

Importing library to split train and test sets

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.25)
```

```
X_train.shape
```

```
(300, 2)
```

```
Y_train.shape
```

```
(300,)
```

The shape function is used to validate if the training and test sets are properly split.

STEP 5:

Libraries are imported to access the function which will be trained on the given data. The library Standard Scaler is used to transform the values of the data which is readable by the function and can be processed.

Example: Male and Female will be converted to 01 and 10 which is readable by the function.

Importing library to tranform train and test sets for implementing KNN

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

The X_train and X_test are transformed and stored in the same variable so that the data is overridden.

STEP 6:

Now the most important step, the KNeighborsClassifier function is imported from sklearn library.

The function KNeighborsClassifier is stored in a variable classifier, function fit is used on the classifier which trains the model.And function predict is used on classifier to predict the values and these predictions are stored in a new variable Y_pred

Import KNeighborsClassifer function to implement the model

```
from sklearn.neighbors import KNeighborsClassifier
```

```
classifier = KNeighborsClassifier(n_neighbors=5, metric ='minkowski')
```

```
classifier.fit(X_train, Y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
Y_pred= classifier.predict(X_test)
```

```
np.array(Y_test)
```

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0], dtype=int64)
```

The 4th part of the data which is Y_test is converted to array to look at the output stored in it, these values will be compared with the values stored in Y_pred.

STEP 7:

Comparing the Y_pred (Predicted values) and the output of the test data i.e. Y_test will be compared, and a matrix will be created this matrix consist of 4 elements namely False positive, False negative, True positive and True negative.

This is generated using confusion_matrix function imported from metrics folder in sklearn library

Importing confusion_matrix function to create confusion matrix

```
from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(Y_test,Y_pred)
```

```
cm
```

```
array([[57,  2],
       [ 6, 35]], dtype=int64)
```

The matrix is stored in a variable and then printed.

STEP 8:

Finally, the accuracy is calculated by importing accuracy_score function from metrics folder in sklearn library.

Importing accuracy_score calculate accuracy

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.92
```

The accuracy of our model is 92%.

Thank You

NICE
Software Solutions

21, IT Park, Gayatri Nagar, Behind VNIT,
Nagpur – 440022

Contact Us:
Mob.: India- +91 8007861579, Dubai- +971 508794909
Web: www.nicesoftwaresolutions.com
E-Mail: info@nicesoftwaresolutions.com