SERVICE CANCELLATION PREDICTOR REPORT

Teaching assistant : Dr . Aya Saad

Team ID : 14

Department: General

Course : Artificial Intelligence

TEAM MEMBERS

ID	NAME
20201700840	1- مصطفي محمود محمد محمد سنبل
20201700259	2- رحمة مدحت عبدالرحمن عبدالحليم
20201701149	3- نسمة فريد محمد عبدالرحمن
20201700918	4- ندي السعيد جابر ملوحه
20201700921	5- ندى عبدالله مهدي علي الدهشوري
20201700254	6- رحاب فاضل فارس فاضل
20201700831	7- مصطفي عاطف محمد هاشم

INTRODUCTION:-

It is that we, as a company, offer offers and services, so we want to expect the customer when he subscribes to a certain service, when can he cancel this subscription, or what would make him think that he cancels the subscription, like for example a certain service that takes a lot of money and does not benefit from it So we are trying to see what services can make the customer leave any service we have, and we are trying to exclude them

Preprocessing :-

- ***Frist we import pandas to make the data clean from null values by (dropna) function***

- ***Delete unwanted feature (customerID)it isn't useful in our prediction***

- ***We find data type of (TotalCharges) is wrong it was object but it must be float we converted its data type to float after delete 10 rows from excel sheet whose have blank cell***
  To be 7033 row instead of 7043 row

- ***Converted categorical values to numerical values in the following columns***
  ['gender', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod']

- **By user_defined function called outliers that take data and features as parameters to calculate extreme values by statistics lows**

```
q1 = df[ft].quantile(0.25)
q3 = df[ft].quantile(0.75)
```

then calculate interquntile range(iqr) to find values that greater than (q3+1.5*iqr) or that less than (q1-1.5*iqr) and put they in list (ls) and return it

```
iqr = q3 - q1
lower = q1 - (1.5) * iqr
upper = q3 + (1.5) * iqr
ls = df.index[(df[ft] < lower) | (df[ft] > upper)]
return ls
```

- **then loop on all features to get indices (rows) that represented outliers**

- **then create user_defined function called remove that take data and list that has indices of outliers**

```
def remove(df, ls):
 ls = sorted(set(ls))
 df = df.drop(ls)
 return df
```

- **then we store new data in :-**

```
data_new = remove(data, index_list)
```

using remove outliers it was very useful to make the accuracy higher than before we use it , it make accuracy of svm=0.78 instead of 0.73

Data scaling:

```
# data scaling
scaler =StandardScaler()
scaler.fit(data_new)
```

It is step of preprocessing that is applied on independent features of data , it help us to normalize data within particular range

We import

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

and this is formula that used in backend

$$z=\frac{x-\mu}{\sigma}$$

Split data :

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

we import from sklearn library train_test_split to split new data to 70% training and 30% is test

- ***and we drop churn from training and test because it the label (output) of prediction.***

Feature Extraction :

A linear or non-linear transform on the original feature space .

$$\begin{bmatrix} X1 \\ \vdots \\ \vdots \\ Xn \end{bmatrix} \Rightarrow \begin{bmatrix} Y1 \\ \vdots \\ \vdots \\ Yn \end{bmatrix} = f\left(\begin{bmatrix} X1 \\ \vdots \\ \vdots \\ Xn \end{bmatrix}\right)$$

Linear discriminant analysis (LDA) :

- ***Supervised feature extraction***
- ***LDA is a traditional statistical technique that reduces dimensionality while preserving as much of the class discriminatory information as possible. The conventional form of the LDA assumes that all the data are available in advance and the LDA feature space is computed by finding the eigendecomposition of an appropriate matrix.***

Linear discriminant analysis Algorithm Equations :

$$J(w) = \frac{|\mu 1 - \mu 2|^2}{S_1^2 + S_2^2}$$

Algorithms that were used in this project :-

1. **Logistic regression.**
2. **SVM.**
3. **Decision Tree (ID3).**
4. **Naïve Bayes.**
5. **Random Forest3.**
6. **KNN.**

(We will explain each one separately.)

1. *Logistic Regression(LR)*

➢ ***Logistic Regression*** → ***is a Classification algorithm used to assign observations to a discrete set of classes***

➢ ***Logistic Regression* Algorithm Equations** :

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

➢ ***The libraries were used in Algorithm***

from sklearn.linear_model import LogisticRegression

➢ ***Accuracy Calculation in (Train and Test)***

- *Accuracy(Train)*
  - ***The function was used in => score***
  - ***Accuracy = 0.8287173971497714***
- *Accuracy(Test)*
  - ***The function was used in => metrics.accuracy_score***
  - ***Accuracy = 0.8012539184952978***

➢ **confusion_matrix** → $\begin{bmatrix} 1077 & 124 \\ 193 & 201 \end{bmatrix}$.

2.Support Vector Machine (SVM)

➢**Support Vector Machine:** is a supervised machine learning model that uses classification algorithms for two-group classification problems.

➢**The type of Kernel Trick we used:** Linear Kernel.

➢**The formula of Linear Kernel :** $K(x,xi)=\sum(x * xi)$.

➢**The libraries were used in Algorithm:** SVC from sklearn.svm.

➢ **Accuracy Calculation in (Train and Test):**

- **Accuracy(Train)**
  - The function was used in →(score).
  - Accuracy = 0.7652594783543963 .
- **Accuracy(Test)**
  - The function was used in →(accuracy_score).
  - Accuracy = 0.7529780564263323.

➢**The Algorithms we tested them to select the features :**

- **Chi2**
  - **Accuracy(Train)**
    - The function was used in →(score).
    - Accuracy = 0.6524096325874125.
  - **Accuracy(Test)**
    - The function was used in → (accuracy_score).
    - Accuracy = 0.6120478369909565.
- **F_classif**
  - **Accuracy(Train)**
    - The function was used in →(score).
    - Accuracy = 0.6427493628574318.
  - **Accuracy(Test)**
    - The function was used in → (accuracy_score).
    - Accuracy = 0.6202478549908655

➢**confusion_matrix.** → $\begin{bmatrix} 1201 & 0 \\ 394 & 0 \end{bmatrix}$.

3.Decision Tree (ID3)

➢ **Decision Tree** : is a structure that contains nodes and edges and is built from a dataset. Each node is either used to make a or represent an outcome.

➢ **ID3 (Iterative Dichotomiser 3)** : is a part of decision tree learning, stands for Iterative Dichotomiser 3, is a classification algorithm that follows a greedy approach of building a decision tree by selecting a best attribute that yields maximum Information Gain (IG) or minimum Entropy (H).

➢ **ID3 Algorithm Equations** :

- **Entropy**

$$H(S) = \sum -p(x) \log p(x)$$

- **Information Gain**

$$IG(S, A) = H(S) - \sum p(t)\, H(t) = H(S) - H(S\backslash A)$$

➢ **The Library that used in Project** :

DecisionTreeClassifier from sklearn.tree .

➢ **Accuracy Calculation in (Train and Test) :**

- **Accuracy (Train) :**
  - o The function was used in → score.
  - o Accuracy = 0.8093573541274536 .
- **Accuracy (Test) :**
  - o The function was used in → accuracy_score.
  - o Accuracy = 0.7918495297805642.

➢**confusion_matrix**. → $\begin{bmatrix} 1127 & 74 \\ 219 & 129 \end{bmatrix}$.

4. K-Nearest Neighbour (KNN)

➢**KNN → stands for K-nearest neighbour, it's one of the Supervised learning algorithm mostly used for classification of data on the basis how it's neighbour are classified. KNN stores all available cases and classifies new cases based on a similarity measure. K in KNN is a parameter that refers to the number of the nearest neighbours to include in the majority voting process.**

➢**The KNN algorithm : is a supervised machine learning model. That means it predicts a target variable using one or multiple independent variables.**

➢**KNN classification algorithm based on →** density .

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}\ .$$

➢**The library was used in Algorithm:-**

KNeighborsClassifier from sklearn.neighbors .

➢**Accuracy Calculation in (Train and Test)**

- **Accuracy(Train)**
  - o The function was used in →score.
  - o Accuracy = 0.8158107018015596.
- **Accuracy(Test)**
  - o The function was used in → accuracy_score.
  - o Accuracy = 0.7943573667711599.

➢confusion_matrix → $\begin{bmatrix} 1055 & 146 \\ 182 & 212 \end{bmatrix}$

5. Random Forest

➢ Random Forest : is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

➢The Random Forest algorithm :

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: **Select random K data points from the training set.**

Step-2: **Build the decision trees associated with the selected data points (Subsets).**

Step-3: **Choose the number N for decision trees that you want to build.**

Step-4: **Repeat Step 1 & 2.**

Step-5: **For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes**.

➢The library was used in Algorithm:-

RandomForestClassifier from sklearn.ensemble.

➢Accuracy Calculation in (Train and Test)

- **Accuracy(Train)**
  - o The function was used in → score .
  - o Accuracy = 0.8830330734068298.

- **Accuracy(Test)**
  - The function was used in → accuracy_score.
  - Accuracy = 0.806269592476489

➢confusion_matrix → $\begin{bmatrix} 1102 & 99 \\ 210 & 184 \end{bmatrix}$.

6.Naïve Bayes (idiot Bayes)

➢**Naive Bayes :** is a classification algorithm for binary (two-class) and multiclass classification problems that works based on the Bayes theorem .

➢ ***Naïve Bayes* Algorithm Equations** :

Bayes theorem ⟹ $P(A\backslash B) = \frac{P(B\backslash A) * P(A)}{P(B)}$.

➢**The library was used in Algorithm:-**

GaussianNB from sklearn.naive_bayes.

➢**Accuracy Calculation in (Train and Test)**

- **Accuracy(Train)**
  - o The function was used in →score.
  - o Accuracy = 0.7687550416778703.
- **Accuracy(Test)**
  - o The function was used in → metrics.accuracy_score.
  - o Accuracy = 0.74858934169279.

➢confusion_matrix → $\begin{bmatrix} 907 & 294 \\ 107 & 287 \end{bmatrix}$.

Comparison between Algorithms

	Logistic Regression	***SVM***	***ID3***	***KNN***	***Random Forest***	***Naïve Bayes***
Before feature extraction						
Accuracy (Train)	**0.8287173971497714**	**0.7652594783543963**	**0.8147351438558752**	**0.8158107018015596**	**0.8830330734068298**	**0.7687550416778703**
Accuracy (Test)	**0.8012539184952978**	**0.7529780564263323**	**0.7918495297805642**	**0.7943573667711599**	**0.806269592476489**	**0.7485893416 9279**
After feature extraction						
Accuracy (Train)	**0.8195751546114547**	**0.8268351707448238**	**0.8287173971497714**	**0.8316751815004033**	**0.847270771712826**	**0.8158107018015596**
Accuracy (Test)	**0.8012539184952978**	**0.8006269592476489**	**0.8018808777429467**	**0.7887147335423198**	**0.8**	**0.7943573667711599**