Machine Learning (P02)

Artificial Intelligence, 2022-23

Nuno Veloso (10411), Augusto Pereira (21136), Duarte Melo (21149)

Introduction

For the second project of this curricular unit (Artificial Intelligence) we were challenged to use multiple machine learning methods to analyze data.

The first dataset we used contains information from more than one million unique patients of COVID-19. It contains information of the current symptoms, status, and the patient's medical history to predict where a patient is at high risk of dying from getting COVID or not.

The second dataset contains information of medical cost for more than 1000 people. It contains information of the genre, BMI, number of children, whether it's a smoker or not, region and health insurance cost.

Automatic classification

The business goal of the automatic classification is to increase efficiency. Automatic classification allows for quick and accurate categorization of data, reducing the need for manual labour and increasing the speed of data processing.

In this project, we used Python as the programming language to develop an automatic classification algorithm that predicts whether a person "will die" from COVID or not based on some parameters passed (age, historical condition, etc.).

As you can see in “Python_Covid_Classifier” folder, in the “Covid-Classifier.ipynb” Python Notebook file, we started by reading the csv “covid_data.csv” as a dataframe, using pandas (view Figure 1).

```
In [2]: import pandas as pd
        import numpy as np
        import datetime
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn import tree
        import joblib

In [3]: df = pd.read_csv('covid_data.csv')
        df
```

Out[3]:

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	ASTHMA	INMSUPR	HIPERTEN
0	2	1	1	1	03/05/2020	97	1	65	2	2	...	2	2	
1	2	1	2	1	03/06/2020	97	1	72	97	2	...	2	2	
2	2	1	2	2	09/06/2020	1	2	55	97	1	...	2	2	
3	2	1	1	1	12/06/2020	97	2	53	2	2	...	2	2	
4	2	1	2	1	21/06/2020	97	2	68	97	1	...	2	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1048570	2	13	2	1	9999-99-99	97	2	40	97	2	...	2	2	
1048571	1	13	2	2	9999-99-99	2	2	51	97	2	...	2	2	
1048572	2	13	2	1	9999-99-99	97	2	55	97	2	...	2	2	
1048573	2	13	2	1	9999-99-99	97	2	28	97	2	...	2	2	
1048574	2	13	2	1	9999-99-99	97	2	52	97	2	...	2	2	

1048575 rows × 21 columns

Figure 1 - Python Code (part 1)

Then, we created a new field called “DIED” that shows the value 1 if the person has died from COVID, or the value 2 if the person has not died from COVID (if the field “DATE_DIED” has a valid date, the person has died from COVID) (view Figure 2).

```
[ ]: df['DIED'] = 1

     df['DIED'][df['DATE_DIED'] == "9999-99-99"] = 2

     df

C:\Users\duart\AppData\Local\Temp\ipykernel_14744\2722679262.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['DIED'][df['DATE_DIED'] == "9999-99-99"] = 2
```

[ ]:

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	INMSUPR	HIPERTENSION	OTH	DIED
0	2	1	1	1	03/05/2020	97	1	65	2	2	...	2	1		1
1	2	1	2	1	03/06/2020	97	1	72	97	2	...	2	1		1
2	2	1	2	2	09/06/2020	1	2	55	97	1	...	2	2		1
3	2	1	1	1	12/06/2020	97	2	53	2	2	...	2	2		1
4	2	1	2	1	21/06/2020	97	2	68	97	1	...	2	1		1
...	...	...	...	...	...	...	...	...	...	...	...	...	...		...
1048570	2	13	2	1	9999-99-99	97	2	40	97	2	...	2	2		2
1048571	1	13	2	2	9999-99-99	2	2	51	97	2	...	2	1		2
1048572	2	13	2	1	9999-99-99	97	2	55	97	2	...	2	2		2
1048573	2	13	2	1	9999-99-99	97	2	28	97	2	...	2	2		2
1048574	2	13	2	1	9999-99-99	97	2	52	97	2	...	2	2		2

1048575 rows × 22 columns

Figure 2 - Python Code (part 2)

After that, we removed the null values from the dataframe to make sure that we were not getting wrong results. Then we created an X dataframe that does not contain the column 'DIED' and an Y dataframe that only contains the column 'DIED', so we could use that in our model (to tell the model that it will predict "DIED" based on the other columns) (view Figure 3).

```
In [5]:
# REMOVE NULL VALUES
df.assign(TOBACCO=np.where(df.TOBACCO == 97, None, df.TOBACCO))
df.assign(TOBACCO=np.where(df.TOBACCO == 99, None, df.TOBACCO))
df.assign(PNEUMONIA=np.where(df.PNEUMONIA == 97, None, df.PNEUMONIA))
df.assign(PNEUMONIA=np.where(df.PNEUMONIA == 99, None, df.PNEUMONIA))
df.assign(INTUBED=np.where(df.INTUBED == 97, None, df.INTUBED))
df.assign(INTUBED=np.where(df.INTUBED == 99, None, df.INTUBED))
df.assign(DIABETES=np.where(df.DIABETES == 97, None, df.DIABETES))
df.assign(DIABETES=np.where(df.DIABETES == 99, None, df.DIABETES))
df.assign(COPD=np.where(df.COPD == 97, None, df.COPD))
df.assign(COPD=np.where(df.COPD == 99, None, df.COPD))
df.assign(HIPERTENSION=np.where(df.HIPERTENSION == 97, None, df.HIPERTENSION))
df.assign(HIPERTENSION=np.where(df.HIPERTENSION == 99, None, df.HIPERTENSION))
df.assign(OBESITY=np.where(df.OBESITY == 97, None, df.OBESITY))
df.assign(OBESITY=np.where(df.OBESITY == 99, None, df.OBESITY))



df = df[['TOBACCO', 'AGE', 'PNEUMONIA', 'INTUBED', 'DIABETES', 'COPD', 'HIPERTENSION', 'OBESITY', 'DIED']]
df_X = df.drop(columns='DIED')
df_Y = df['DIED']
```

Figure 3 - Python Code (part 3)

Afterwards, we divided the dataframe in two sets, one for training and the other for testing - we shuffled the dataset to remove any ordering it had and set the test size to 20% (view Figure 4).

```
In [6]:
X_train, X_test, Y_train, Y_test = train_test_split(df_X, df_Y, test_size=0.2, random_state=42, shuffle=True)
#train
#test
```

Figure 4 - Python Code (part 4)

We then used the algorithm "DecisionTreeClassifier()" from "scikit learn" to create our classifying model. Succeeding that, we saved the model as a joblib file to use it later without having to train it again and then calculated its accuracy, getting a 95% accuracy result (view Figure 5).

```
In [8]:
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)

tree.export_graphviz(model, out_file='covid-classifier.dot',
                     feature_names=['TOBACCO', 'AGE', 'PNEUMONIA', 'INTUBED', 'DIABETES', 'COPD', 'HIPERTENSION', 'OBESITY'],
                     class_names=None,
                     label='all',
                     rounded=True,
                     filled=True)

# exportar para tree, para visualizar depois

joblib.dump(model, 'covid-classifier.joblib') # guardar o  model


predictions = model.predict(X_test)

score = accuracy_score(Y_test, predictions)
score

Out[8]:
0.9444770283479961
```

Figure 5 - Python Code (part 5)

As you can see in Figure 6, the model says that if someone is 40 years old, doesn't smoke neither has diabetes, COPD, hypertension, or obesity but is intubated will probably die from COVID.

```
In [9]: model = joblib.load('covid-classifier.joblib') # use saved model (only do this after running it once!)
        predictions = model.predict([[2, 40, 2, 1, 2, 2, 2, 2]]) # predict
        predictions # show predicting (1 - DIED, 2 - NOT DIED)

        C:\Users\duart\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:409: UserWarning: X does not have vali
        d feature names, but DecisionTreeClassifier was fitted with feature names
          warnings.warn(

Out[9]: array([1], dtype=int64)
```

Figure 6 - Python Code Results

Now, if we test the model with someone that is 40 years old, is not intubated but smokes, it says that the person will probably not die from COVID (view Figure 7).

This means that the "intubated" field makes a lot of impact in the result.

```
In [10]: model = joblib.load('covid-classifier.joblib') # use saved model (only do this after running it once!)
         predictions = model.predict([[1, 40, 2, 2, 2, 2, 2, 2]]) # predict
         predictions # show predicting (1 - DIED, 2 - NOT DIED)

         C:\Users\duart\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:409: UserWarning: X does not have vali
         d feature names, but DecisionTreeClassifier was fitted with feature names
           warnings.warn(

Out[10]: array([2], dtype=int64)
```

Figure 7 - Python Code Results

If we change the age to 70 but keep all the other fields the same, the model says that the person will probably die from COVID (view Figure 8).

```
In [11]: model = joblib.load('covid-classifier.joblib') # use saved model (only do this after running it once!)
         predictions = model.predict([[1, 70, 2, 2, 2, 2, 2, 2]]) # predict
         predictions # show predicting (1 - DIED, 2 - NOT DIED)

         C:\Users\duart\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:409: UserWarning: X does not have vali
         d feature names, but DecisionTreeClassifier was fitted with feature names
           warnings.warn(

Out[11]: array([1], dtype=int64)

In [ ]:
```

Figure 8 - Python Code Results

Clustering

The business goal to be achieved, with the use of the clustering technique, is the segmentation where a large and diverse dataset can be segmented into smaller, more manageable groups of similar data.

The dataset used in this topic includes information on individuals' smoking status, BMI, and age, as well as their insurance costs (we removed fields that don't make sense to include in the clustering algorithm). The household number and region columns were discarded as they did not have direct impact on the insurance cost. The clustering algorithm used was K-means, which groups data points into a specified number of clusters. The optimal number of clusters (2) was determined using the silhouette tool in the program.

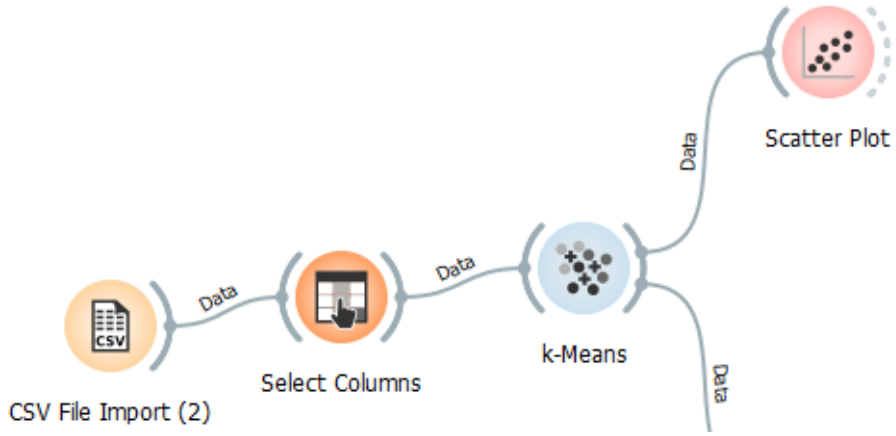

Figure 9 - Orange Clustering

Association rules

The goal of this analysis was to identify patterns and associations in a dataset of insurance charges, to better understand which factors are most strongly associated with higher charges. The dataset included the following fields: BMI, Age, Smoker, and charges (we removed the other fields because they were not relevant for the association rules). The Apriori algorithm was used to mine frequent item sets and association rules, and the results were evaluated based on support and confidence (you can see the association rules made in Orange in the Figure 10).

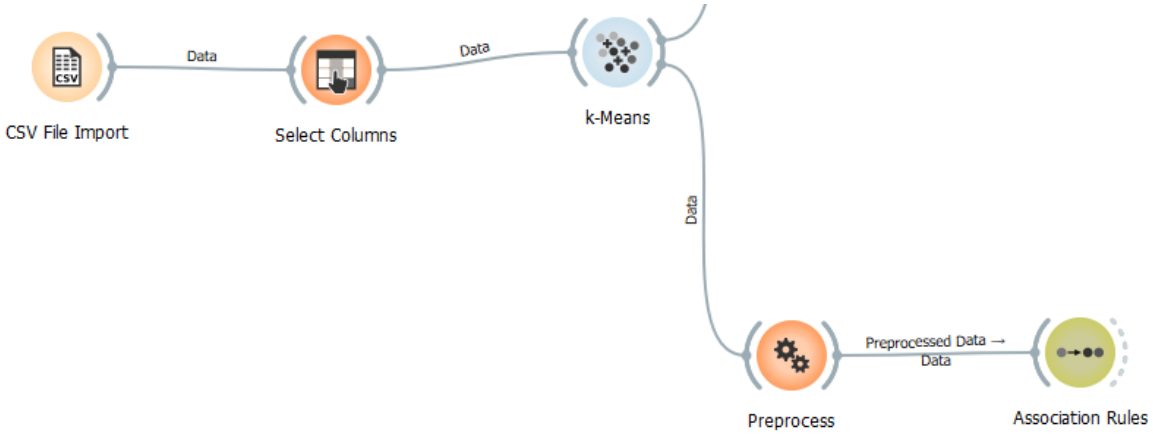

Figure 10 - Association Rules

Results Analysis

For the Python classifier example, through many tests (using different percentages for test and train sets) and accuracy calculations we could understand that the model we developed is really accurate, with a 95% accuracy result as you could see in the “Automatic Classification” topic.

For the clustering algorithm, using the “Orange” software, at first, we tried to use the same dataset (COVID) from the Python example, but then we got bad results. To fix that, we picked another dataset (health insurance costs) that showed us a great clustering graphic and we could retrieve some conclusions from it.

The results of the clustering in the new dataset analysis showed that there is a clear relationship between smoking status, BMI, and age, and insurance costs. Individuals who smoke and have a higher BMI tend to have higher insurance costs, as do older individuals. The clustering also revealed that non-smokers with lower BMI and younger individuals tend to have lower insurance costs.

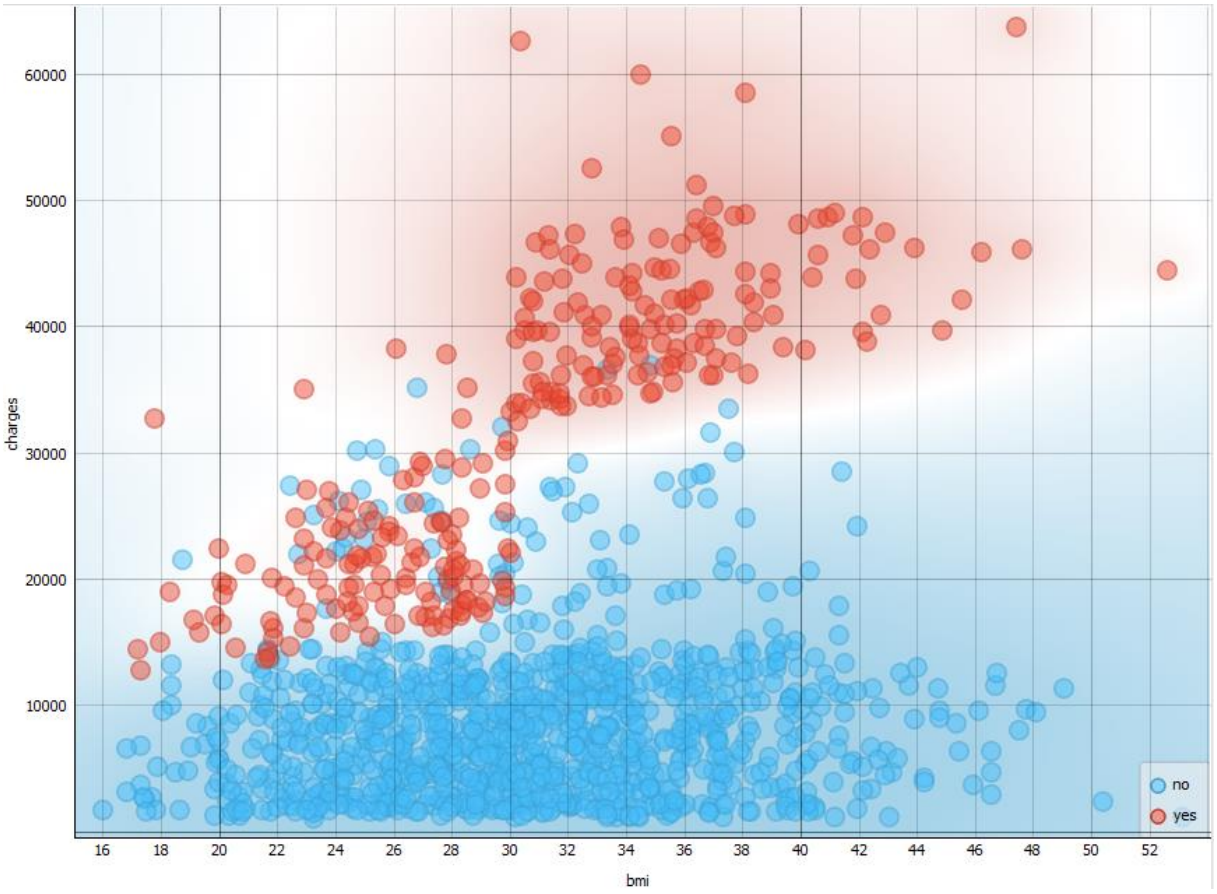

Figure 11 - Cluster Graph

As you can see in Figure 11, this was the clustering graphic we got from using the K-Means algorithm on the health insurance dataset and it shows the relation between the insurance price charged to an individual depending on their BMI and if the person smoked or not, and as you can see if the individual is a smoker the price exponentially rises.

For the association rules, we could get some rules that could give us some insights about how smoking and having a high BMI would assign to the individual a high health insurance cost.
With the use of the algorithm some of the association rules were discovered, such as:

- Someone who's charge is less than 4729$, is less likely to be a nonsmoker with a support of 0.250 and a confidence of 1.
- Someone who is charged with a value between 9382.02$ and 16622.1$ is likely to be a nonsmoker with a support of 0.235 and a confidence of 0.943.
- Someone who is a smoker is likely to be charged more than 16622.1$ with a support of 0.191 and a confidence of 0.931
- Someone who's BMI is higher than 34.6875 and is a smoker is likely to be charged with more than 16622.1$ with a support of 0.058 and a confidence of 1
- Someone who's age is between 39.5 and 51.5 and is also a smoker is likely to be charged 16622.1$ with a support of 0.054 and a confidence of 1.

These are just some of the results obtained through the association rules widget from Orange data mining software.

With these results we can conclude that high values in age, BMI and a yes when it comes to smoking habits will have a much higher chance to be associated with high charges.

We can also conclude from the rest of the data, that smoking will negatively affect the outcome regarding the charges made.

Conclusion

This project has demonstrated the use of machine learning techniques to analyze data from COVID-19 patients and health insurance costs. We have been able to identify patterns and relationships, within the datasets, that may provide valuable insights into the characteristics of two different datasets and potential results/predictions. However, it is important to note that the findings of this study are based on two specific datasets and may not be generalizable to other populations.

GitHub Repository

The GitHub repository for this project can be found in the next url: https://github.com/a10411/IAP02