Politécnico de Coimbra

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Introdução à Inteligência Artificial

Ano Letivo 2021/22

Trabalho 2: Machine Learning

Elaborado em: 2022/01/30

Daniel Santos | Nº2019133865

Assinado por: **DANIEL ALEXANDRE MENDES SANTOS**
Num. de Identificação: 30492115
Data: 2022.01.30 23:52:30+00'00'

CHAVE MÓVEL

Índice

List of Figures

1. Introdução

Este trabalho foi realizado no âmbito da disciplina de Introdução à Inteligência Artificial por Daniel Santos a pedido de trabalho do professor Luís Veloso. Neste trabalho aprendi a trabalhar com o Orange Data Mining e diversas bibliotecas de Python.

No Python (PyCharm IDE) irei demonstrar como formatar/analisar dados, realizar testes e analisar os seus resultados através de gráficos.

No programa Orange Data Mining (ou só Orange), irei aprofundar as funcionalidades dos widgets e analisar os seus resultados.

O relatório foi desenvolvido através da descrição do que fui fazendo e com o devido acompanhamento de screenshots.

2. Secção I – Data wrangling (munging) and Exploration (Python)

Nesta secção comecei primeiro por transformar o ficheiro fornecido .csv numa variável para manipular os dados do dataset AirlinePassangerSatisfaction.csv.

Depois comecei por fazer uma análise das bibliotecas necessárias para suportar o código da secção 1 (pandas, numpy, seaborn, matplotlib, scipy).

A seguir comecei por fazer um menu onde indica todos os passos da ficha:

```
print("\nMenu:")
print("0. Sair")

print("Aquisição de dados:")
print("1. Visualizar dataset\n2. Visualizar as 15 prime
print("5. Adicionar os mesmos headers que retirou anter

print("Estatística Descritiva:")
print("9. Obter a estatística descritiva para cada colu
print("13. Apresentar a frequência de cada um dos valor
print("17. Apresentar em caixa de bigodes(boxplot) a re

print("Missing Values:")
print("18. Substituir na coluna Departure Delay in Minu
print("22. Eliminar os missing values da coluna Arrival

print("Normalização:")
print("23. Caracterizar os tipos de dados utilizados no
print("27. Normalizar as colunas Departure Delay in Sec

print("Binning:")
print("28. Criar uma nova coluna (feature) designada po

print("Estatística Inferencial:")
print("31. Obter o coeficiente de correlação entre as d
print("34. Apresentar graficamente a pivot table da alí
```

Figura 1 - Menu secção 1

2.1. Aquisição de dados

Na aquisição de dados o código é bastante simples:

```
elif opcao == "1":
    print(df_train)

elif opcao == "2":
    print(df_train.head(15))

elif opcao == "3":
    print(df_train.shape)

elif opcao == "4":
    df_train = df_train.rename(columns=df_train.iloc[0]).drop(df_train.index[0])

elif opcao == "5":
    df_train = pd.read_csv("train.csv", names=headers, low_memory=False)
    df_train = df_train.iloc[1:]
    df_train.reset_index(drop=True, inplace=True)

elif opcao == "6":
    print(df_train.columns)

elif opcao == "7":
    df_train.rename(columns={'Food and drink': 'Food'}, inplace=True)

elif opcao == "8":
    df_train.index.name = 'Indice'
```

Figura 2 - Aquisição de dados

O código quando executado permite visualizar o conteúdo do dataset, o número de colunas e linhas, remover a linha do cabeçalho, adicionar a linha do cabeçalho, mostrar o conteúdo do cabeçalho, substituir um nome do cabeçalho e indicar o nome do cabeçalho (neste caso, “Indice”).

2.2. Estatística Descritiva

Nesta secção aprendi a obter valores estatísticos e a criar um gráfico (histograma e boxplot):

```
elif opcao == "10":
    print(df_train.describe(include=['object']))

elif opcao == "11":
    print(df_train[['Flight Distance']].describe())

elif opcao == "12":
    print(df_train['Departure Delay in Minutes'].min())
    print(df_train['Departure Delay in Minutes'].max())

elif opcao == "13":
    print(df_train['Class'].value_counts())

elif opcao == "14":
    print(df_train['Class'].value_counts().to_frame())

elif opcao == "15":
    plt.hist(df_train["Departure Delay in Minutes"])
    plt.xlabel("Departure Delay in Minutes")
    plt.ylabel("count")
    plt.title("Histogram")
    plt.show()

elif opcao == "16":
    print(df_train['satisfaction'].unique())

elif opcao == "17":
    sns.boxplot(x="satisfaction", y="Departure Delay in Minutes", data=df_train)
    plt.show()
```

Figura 3 - Estatística Descritiva

O código quando executado permite visualizar a média, desvio padrão, quartis de tudo ou apenas das variáveis categóricas. Obter valores máximos, mínimos, a frequência de valores de uma coluna, como criar um gráfico histograma, obter os valores de uma coluna e por fim visualizar um boxplot entre as variáveis satisfaction e departure delay in minutes.

2.3. Missing Values

Nesta secção aprendi a visualizar e substituir valores nulos:

```
elif opcao == "18":
    df_train['Departure Delay in Minutes'] = df_train['Departure Delay in Minutes'].replace(70, np.NaN)
    df_train['Arrival Delay in Minutes'] = df_train['Arrival Delay in Minutes'].replace(100, np.NaN)
    print(df_train['Departure Delay in Minutes'].head())
    print(df_train['Arrival Delay in Minutes'].head())

elif opcao == "19":
    missing_data = df_train.isnull().sum()
    print(missing_data)

elif opcao == "20":
    print("Missing values")

elif opcao == "21":
    avg_norm_loss = df_train["Departure Delay in Minutes"].astype("float").mean(axis=0)
    df_train["Departure Delay in Minutes"].replace(np.nan, avg_norm_loss, inplace=True)
    print(df_train['Departure Delay in Minutes'].head())

elif opcao == "22":
    df_train.dropna(subset=["Arrival Delay in Minutes"], axis=0, inplace=True)
    df_train.reset_index(drop=True, inplace=True)
    print(df_train['Arrival Delay in Minutes'].head())
```

Figura 4 - Missing Values

O código quando executado permite substituir determinados valores por “NaN”, obter o número de missing values em cada coluna, substituir os missing values pela média da coluna e por fim eliminar os missing values.

Existe várias formas de lidar com os missing values sendo as 3 principais formas: substituir, eliminar e ignorar.

Podemos substituir os missing values, ou através da média/mediana da sua coluna, ou prever qual será o seu valor e substituir.

Podemos eliminar a linha em que o missing value existe.

Podemos usar algoritmos que nos permitam ignorar os missing values.

2.4. Normalização

Nesta secção aprendi a obter os tipos de dados de cada coluna e a limitar os valores que as colunas podem "flutuar":

```
elif opcao == "23":
    print("Caracterizar os tipos de dados utilizados nos dataframes do Pandas")

elif opcao == "24":
    print(df_train.dtypes, df_train.info())

elif opcao == "25":
    varint = ['Flight Distance','Inflight wifi service','Departure/Arrival time convenient', 'Ease of Online booking','Gate locat
    df_train[varint] = df_train[varint].astype("float")

elif opcao == "26":
    df_train['Departure Delay in Seconds'] = df_train['Departure Delay in Minutes'] * 60
    df_train['Arrival Delay in Seconds'] = df_train['Arrival Delay in Minutes'] * 60
    print(df_train['Arrival Delay in Minutes'].head())
    print(df_train['Arrival Delay in Seconds'].head())

elif opcao == "27":
    df_train['Departure Delay in Seconds'] = df_train['Departure Delay in Seconds']/df_train['Departure Delay in Seconds'].max()
    df_train['Arrival Delay in Seconds'] = df_train['Arrival Delay in Seconds'] / df_train['Arrival Delay in Seconds'].max()
    print(df_train['Arrival Delay in Seconds'].head())
```

Figura 5 - Normalização

O código quando executado permite obter os tipos de dados, converter o tipo das colunas (neste caso, int para float), transformar minutos em segundos e delimitar valores das colunas.

Existe 7 tipos de dados nos dataframes do pandas:

Object - Texto ou valores numéricos e não numéricos mistos

Int64 - Números inteiros

float64 - Números de ponto flutuante

bool - Valores verdadeiro/falso

datetime64 - Valores de data e hora

timedelta[ns] - Diferenças entre duas datas

category - Lista finita de valores de texto

2.5. Binning

Nesta secção aprendi a categorizar e a comparar colunas:

```
elif opcao == "28":
    bins = np.linspace(min(df_train['Departure Delay in Seconds']), max(df_train['Departure Delay in Seconds']), 4)
    group_names = ['Low', 'Medium', 'High']
    df_train['Departure Delay in Seconds_Binned'] = pd.cut(df_train['Departure Delay in Seconds'], bins, labels=group_names,

elif opcao == "29":
    print(df_train[['Departure Delay in Seconds','Departure Delay in Seconds_Binned']].head(5))

elif opcao == "30":
    plt.hist(df_train["Departure Delay in Seconds_Binned"])
    plt.xlabel("Departure Delay in Seconds_Binned")
    plt.ylabel("count")
    plt.title("Histogram")
    plt.show()
```

Figura 6 - Binning

O código quando executado permite categorizar os valores dos atrasos em 3 categorias (baixa, média e elevada) e comparar a nova coluna criada (categorizada) com a velha (numeral).

2.6. Estatística Inferencial

Nesta secção aprendi a obter coeficientes e a criar vários tipos de tabelas e gráficos:

```
elif opcao == "31":
    print(df_train.corr())

elif opcao == "32":
    sns.regplot(x="Departure Delay in Minutes", y="Arrival Delay in Minutes", data=df_train)
    plt.ylim(0, )
    plt.show()

elif opcao == "33":
    df_new = df_train[['Age', 'Type of Travel', 'Class']]
    grouped = df_new.groupby(['Type of Travel', 'Class'], as_index=False).mean()
    grouped_pivot = grouped.pivot(index='Type of Travel', columns='Class')
    print(grouped_pivot)

elif opcao == "34":
    plt.pcolor(grouped_pivot, cmap='RdBu')
    plt.colorbar()
    plt.show()

elif opcao == "35":
    df_groups = df_train[['Age', 'satisfaction']]
    grouped_by_mean = df_groups.groupby(['satisfaction'], as_index=False).mean()
    print(grouped_by_mean)

elif opcao == "36":
    grouped = df_groups.groupby(['satisfaction'])
    df_groups['Age'].unique()
    f_val, p_val = f_oneway(grouped.get_group('neutral or dissatisfied')['Age'], grouped.get_group('satisfied')['Age'])
    print(f_val,p_val)
```

Figura 7 - Estatística Inferencial

O código quando executado permite obter o coeficiente de correlação entre as features do dataset, obter o scatterplot das colunas com maior coeficiente de correlação, obter uma pivot table, apresentar a pivot table como heatmap, agrupar valores de uma só coluna e usar o one-way ANOVA.

3. Secção II – Machine Learning (Orange)

Esta secção do trabalho usou o programa Orange com os datasets train.csv e test.csv.

Foi usado como target a variável “gender”.

A matriz de confusão é uma tabela comparativa de valores que um algoritmo previu em relação aos valores reais ocorridos.

Ou seja, depois de treinar um modelo binário de machine learning e aplicar as previsões sobre o conjunto de dados separado para teste, o resultado obtido será expresso em uma coluna com as previsões:

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Figura 8 - Matriz de confusão

As curvas ROC (parece graficamente uma montanha) são uma forma de representar a relação entre a sensibilidade e a especificidade de um teste diagnóstico quantitativo. Descrevem a capacidade discriminativa de um teste diagnóstico para um determinado número de valores "cutoff point". Isto permite pôr em evidência os valores para os quais existe maior optimização da sensibilidade em função da especificidade.

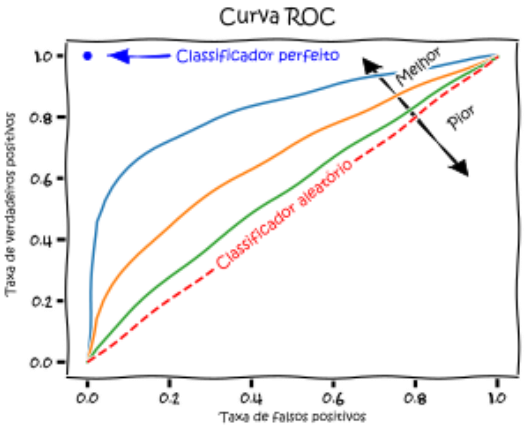

Figura 9 - Curva ROC

Existe várias métricas que irei analisar ao longo do programa Orange: AUC, CA, F1, Precision e Recall.

AUC significa “Area under the ROC curve”. Esta calcula a área debaixo da curva ROC:

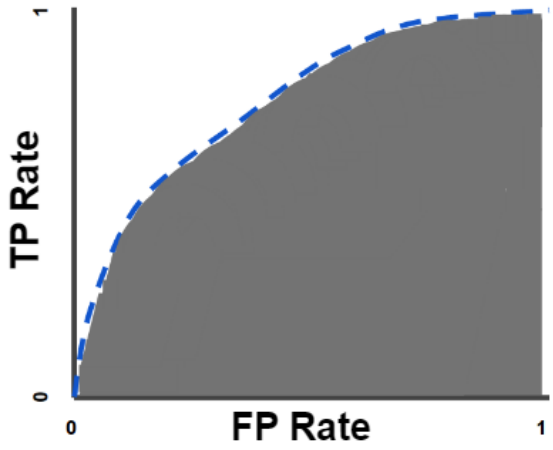

Figura 10 - Curva ROC e AUC

CA indica a fração de previsões que o algoritmo acertou.

F1 indica a média ponderada entre a métrica precisão e recall.

Precision verifica que proporção de previsões estão realmente corretas.

Recall verifica que proporção de previsões foram identificadas corretas.

Agora que está explicado o significado das métricas irei visualizar e analisar o widget predictions:

Model	AUC	CA	F1	Precision	Recall
kNN	0.499	0.499	0.498	0.498	0.499
Logistic Regression	0.501	0.508	0.400	0.507	0.508
Tree	0.553	0.595	0.594	0.595	0.595
SVM	0.505	0.500	0.389	0.478	0.500
Random Forest	0.668	0.595	0.595	0.595	0.595

Figura 11 - Prediction (secção2)

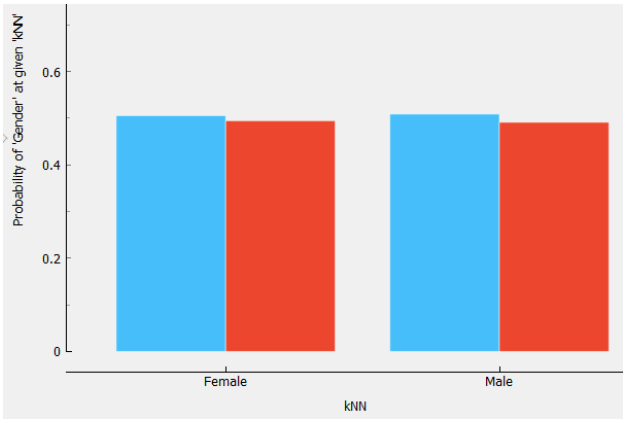

Figura 12 - Distribution KNN (secção2)

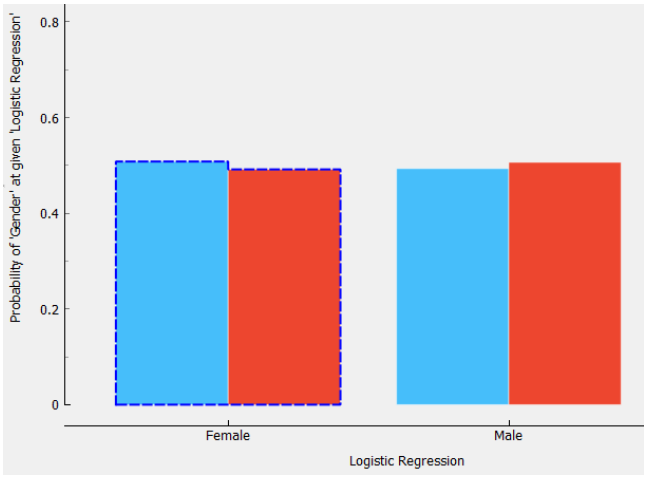

Figura 13 - Distribution Logistic Regression (secção2)

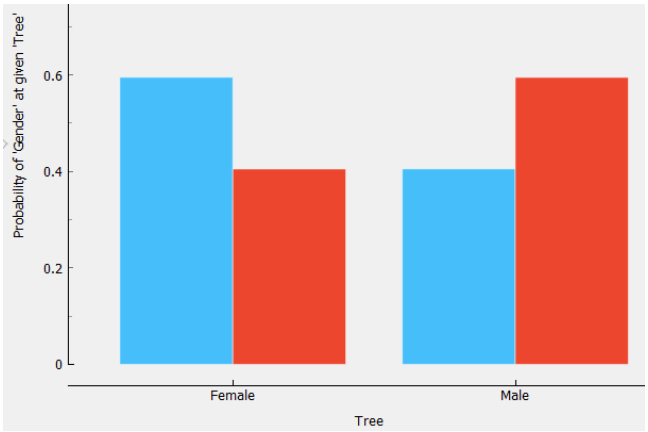

Figura 14 - Distribution Tree (secção2)

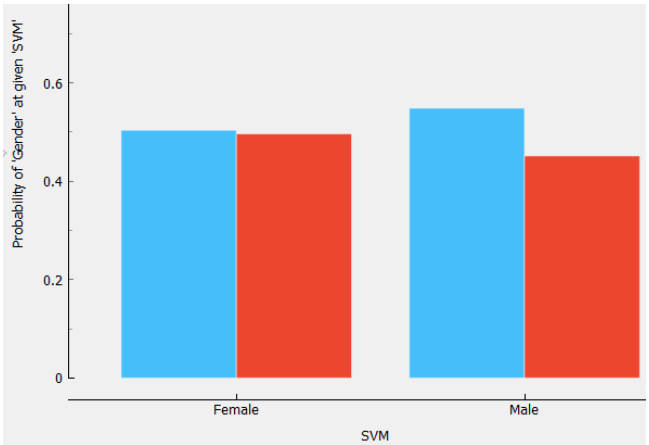

Figura 15 - Distribution SVM (secção2)

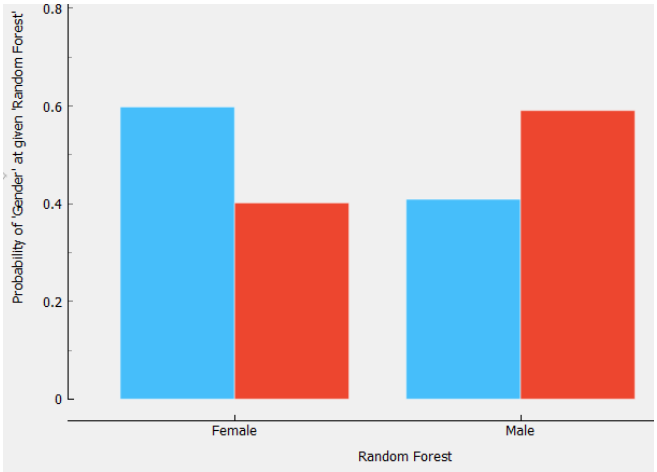

Figura 16 - Distribution Random Forest (secção2)

Como podemos observar os algoritmos tiveram uma precisão por volta dos 50%, sendo que o tree e o random forest tiveram uma maior variação de previsões, mas enquanto que a tree tem em média uma previsão de 50% a random forest tem uma previsão de 60%.

Para conseguir apenas tirar os dados errados fui à confusion matrix e selecionei apenas os dados errados:

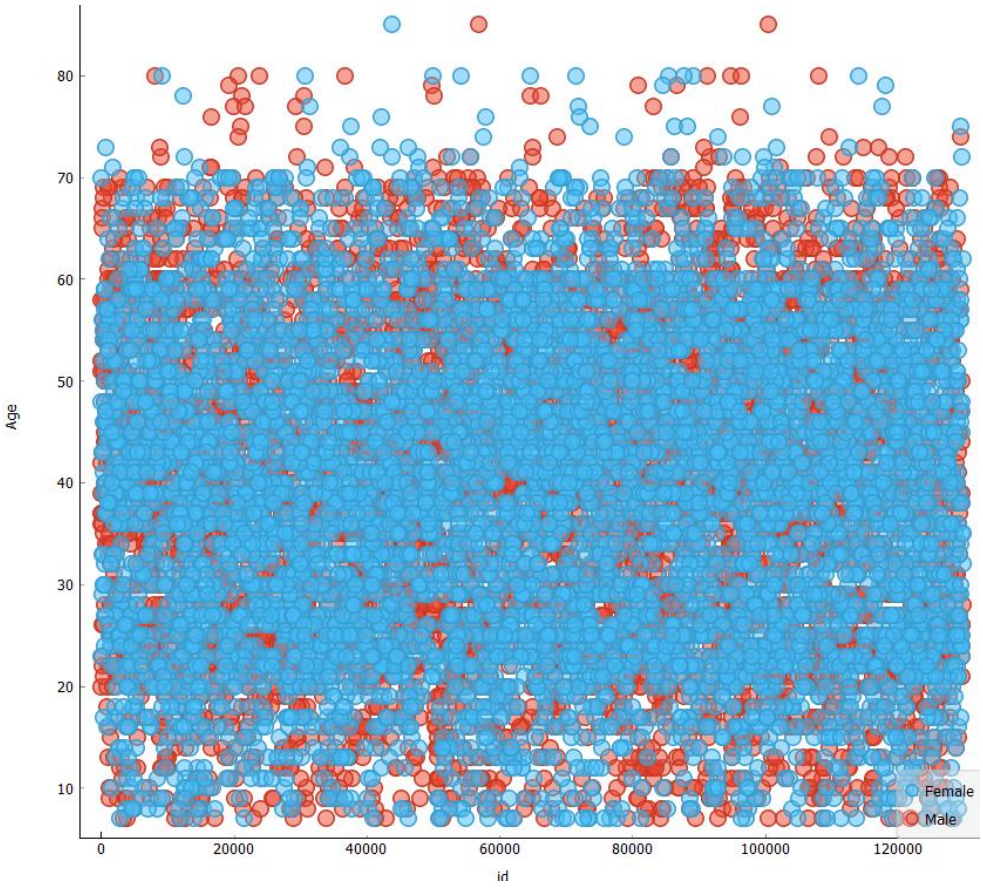

Figura 17 - Scatter Plot dadoserrados (secção2)

Como podemos observar o predictions assume que as mulheres começam a andar de avião mais cedo, mas os homens andam de avião até mais velhos.

Agora vou passar a fazer Test and score com cross validation:

Evaluation Results

Model	AUC	CA	F1	Precision	Recall
kNN	0.496	0.497	0.497	0.497	0.497
Tree	0.549	0.592	0.592	0.592	0.592
SVM	0.499	0.502	0.387	0.483	0.502
Random Forest	0.666	0.595	0.595	0.595	0.595
Logistic Regression	0.500	0.507	0.369	0.499	0.507

Model Comparison by AUC

	kNN	Tree	SVM	Rando...	Logisti...
kNN		0.000	0.353	0.000	0.156
Tree	1.000		1.000	0.000	1.000
SVM	0.647	0.000		0.000	0.386
Random Forest	1.000	1.000	1.000		1.000
Logistic Regression	0.844	0.000	0.614	0.000	

Figura 18 - Test and Score CrossValidation (secção2)

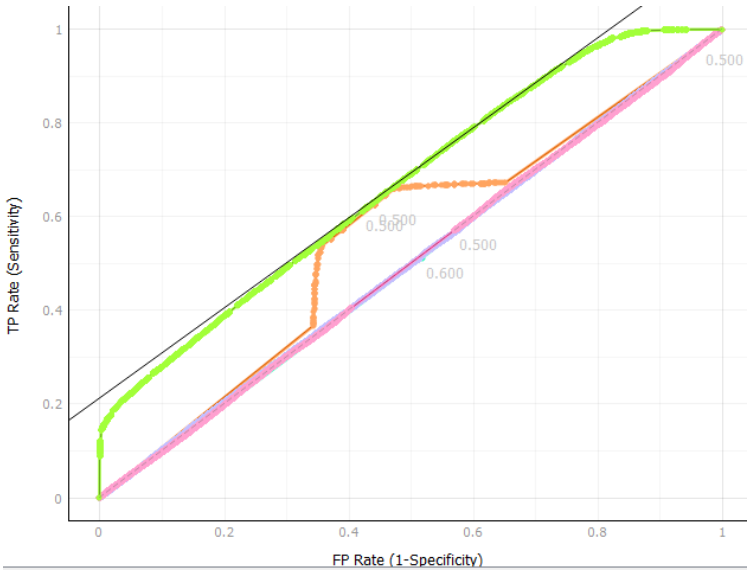

Figura 19 – ROC CrossValidation (secção2)

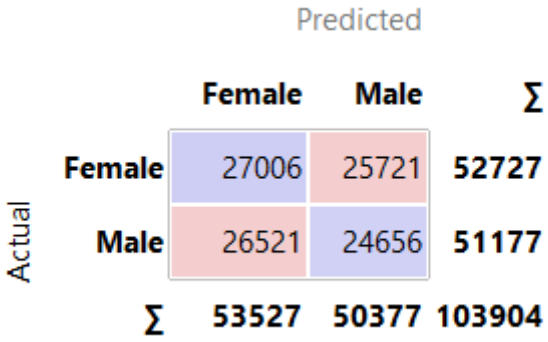

Figura 20 - Confusion Matrix KNN (secção2)

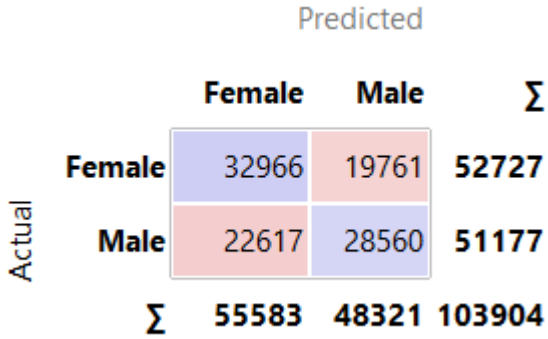

Figura 21 - Confusion Matrix Tree (secção2)

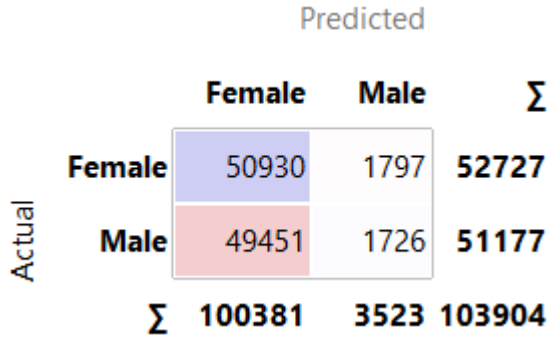

Figura 22 - Confusion Matrix Logistic Regression (secção2)

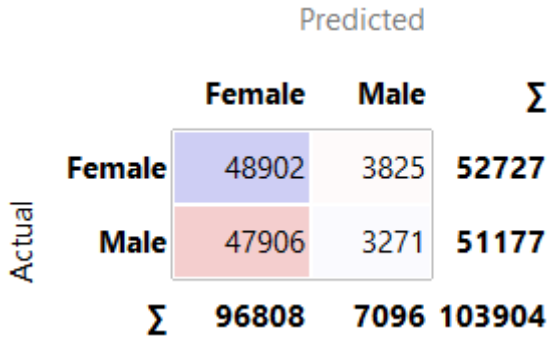

Figura 23 - Confusion Matrix SVM (secção2)

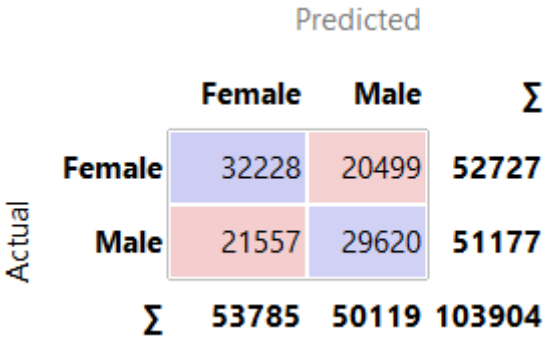

Figura 24 - Confusion Matrix Random Forest (secção2)

Este método de cross validation deu aos algoritmos uma previsão correta de 50% dos dados, sendo que, a tree e a random forest continuam a ser as que mais se destacam como podemos observar na curva ROC.

Agora vou passar a fazer Test and score com test on train data:

Model	AUC	CA	F1	Precision	Recall
kNN	0.743	0.684	0.684	0.684	0.684
Tree	0.994	0.950	0.950	0.951	0.950
SVM	0.486	0.501	0.386	0.480	0.501
Random Forest	0.997	0.974	0.974	0.974	0.974
Logistic Regression	0.501	0.507	0.374	0.501	0.507

Figura 25 - Test and Score Test on train data (secção2)

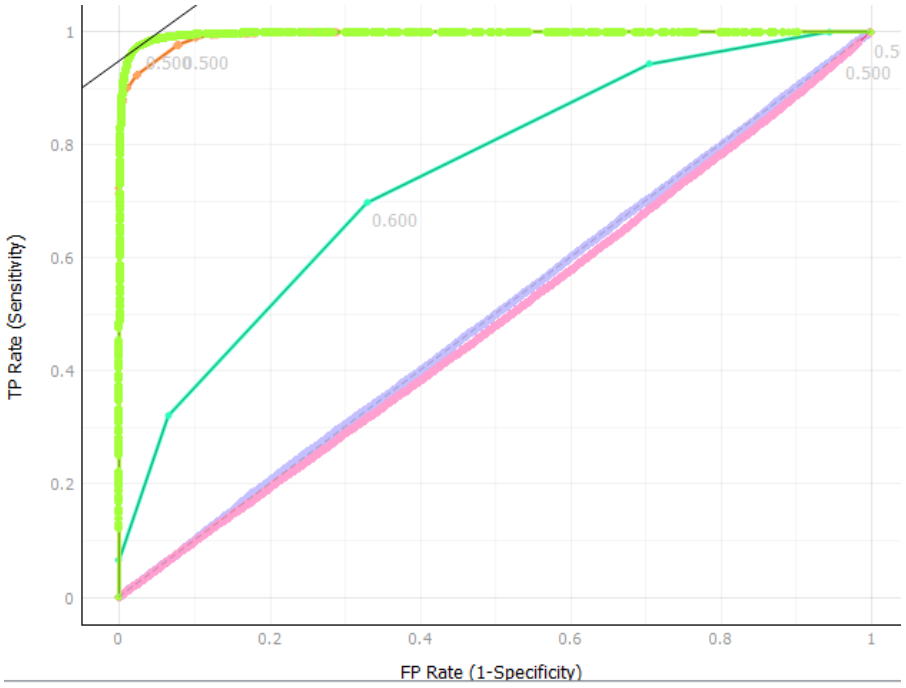

Figura 26 - ROC Test on train data (secção2)

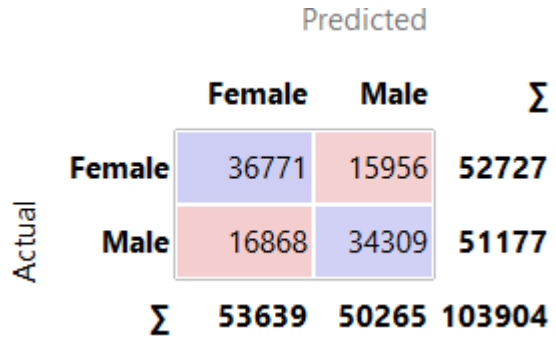

Figura 27 - Confusion Matrix KNN (secção2)

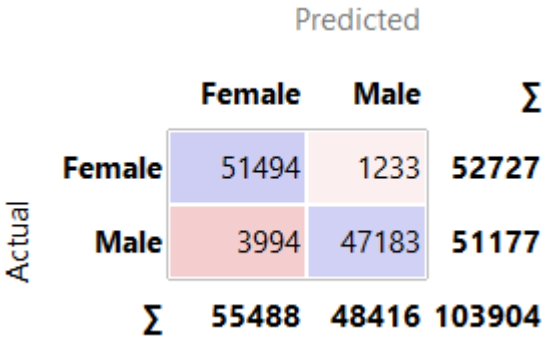

Figura 28 - Confusion Matrix Tree (secção2)

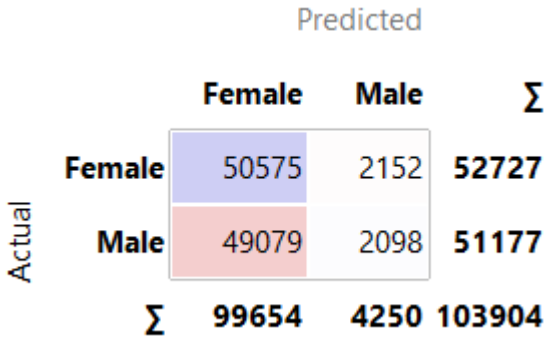

Figura 29 - Confusion Matrix Logistic Regression (secção2)

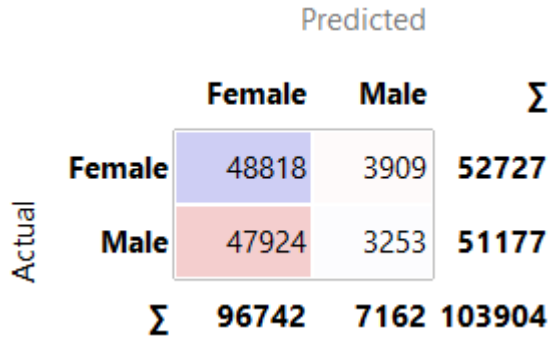

Figura 30 - Confusion Matrix SVM (secção2)

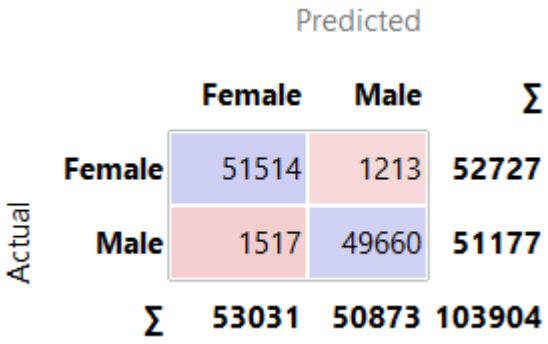

Figura 31 - Confusion Matrix Random Forest (secção2)

Como se pode observar quem teve melhor prestação foi os algoritmos Logistic Regression e o SVM visto que estes conseguiram acertar cerca de 50% dos casos. Os outros 3 algoritmos não se conseguiram adaptar corretamente pois ficaram overfit e não souberam classifar os dados corretamente.

A seguir usei o widget Rank para obter as features mais importantes:

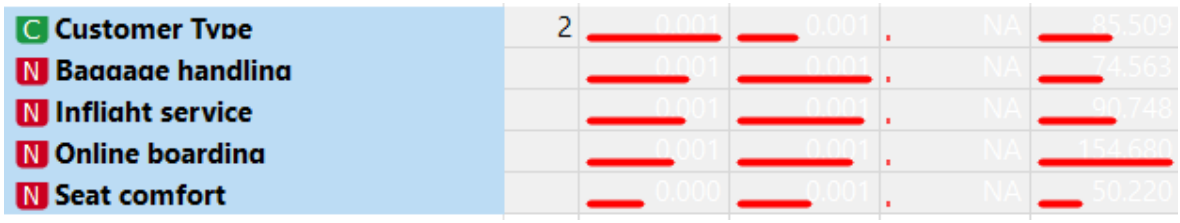

Figura 32 - Rank (secção2)

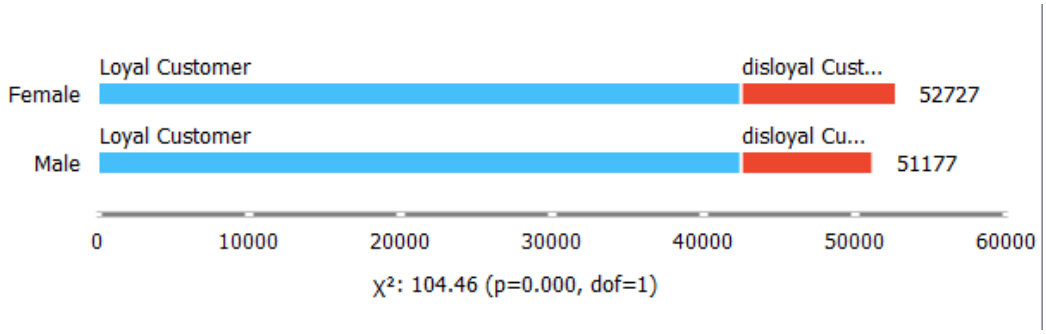

Figura 33 - Boxplot (secção2)

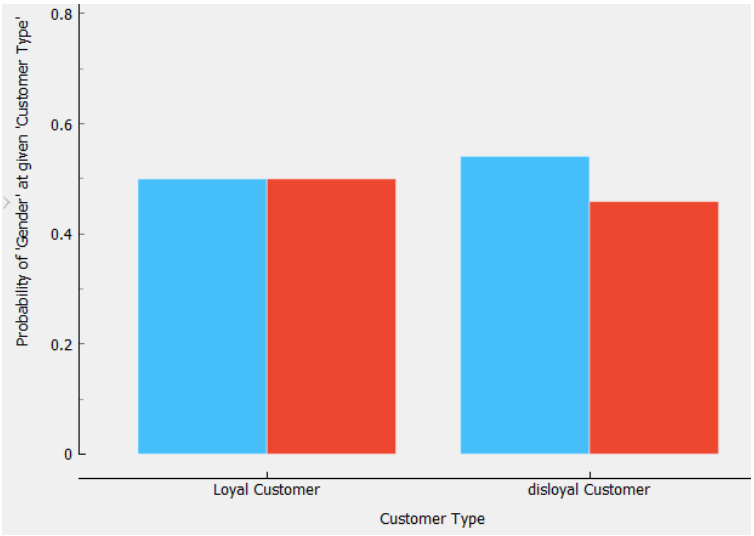

Figura 34 - Distributions Rank (secção2)

Como podemos observar a maior relação com o gênero é o customer type e podemos concluir que, apesar de uma variação pequena, uma mulher (descontente ou não) será mais leal que um homem para a companhia aérea.

Ao selecionar apenas as features que estiveram top 5 e recorrendo ao score Silhouette para definir o K do K-means observei:

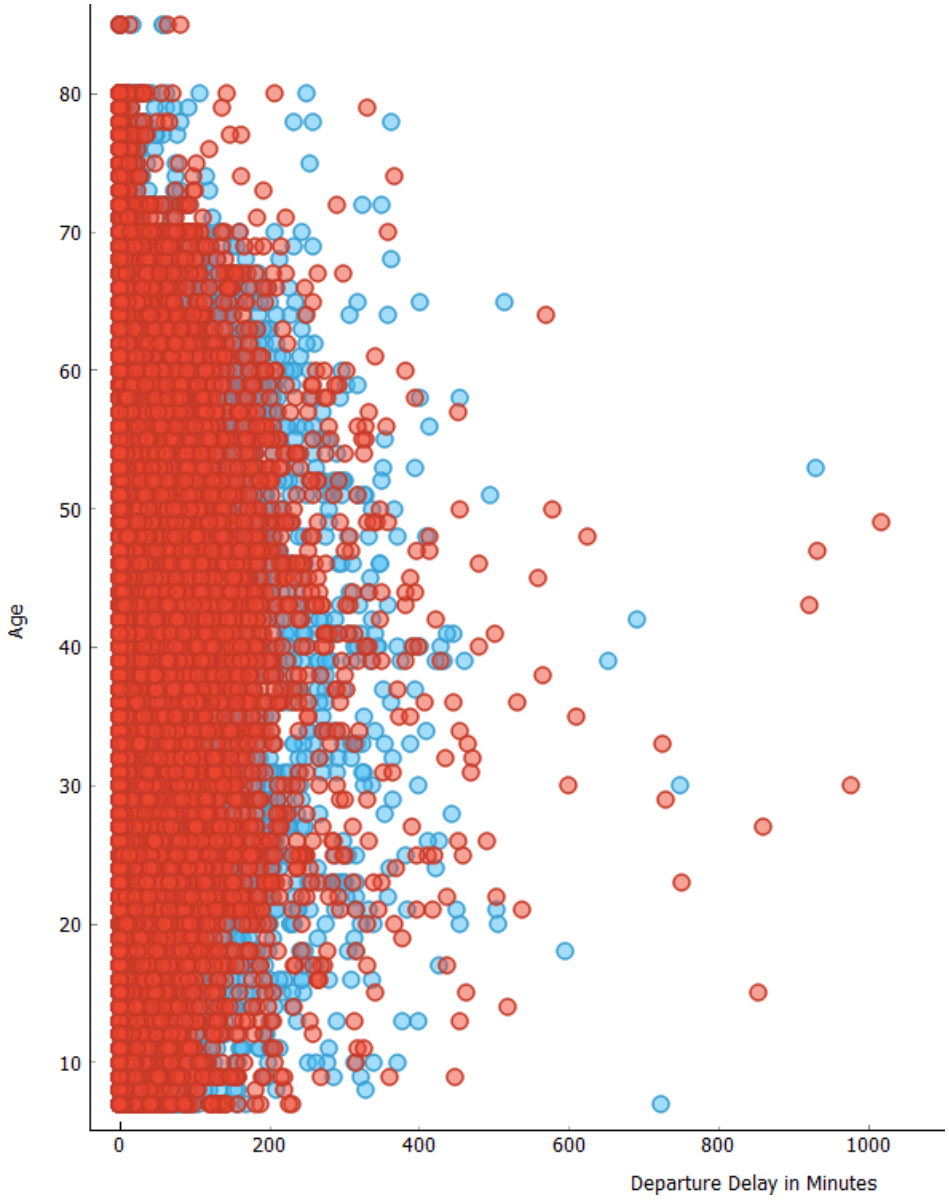

Figura 35 - Scatterplot Rank (secção2)

Como se pode observar as mulheres têm uma tendência maior a apanhar atrasos nas linhas de voo.

Agora recorri ao widget distance para usar o algoritmo hierichical clustering e observei no scatterplot:

Figura 36 - Scatterplot HC (secção2)

Como se pode observar o gráfico está bastante parecido ao anterior o que mostra que o gênero não tem grande variação com as variáveis selecionadas através do rank.

4. Secção III – Machine Learning (Orange)

Neste dataset Stroke.csv irei analisar as variáveis com target à feature Stroke.

Primeiro comecei por analisar o widget predictions:

Model	AUC	CA	F1	Precision	Recall
Random Forest	0.998	0.975	0.971	0.976	0.975
SVM	0.580	0.917	0.912	0.908	0.917
Logistic Regression	0.808	0.949	0.928	0.915	0.949
Tree	0.500	0.951	0.928	0.905	0.951
kNN	0.942	0.952	0.930	0.941	0.952

Figura 37 - Prediction (secção3)

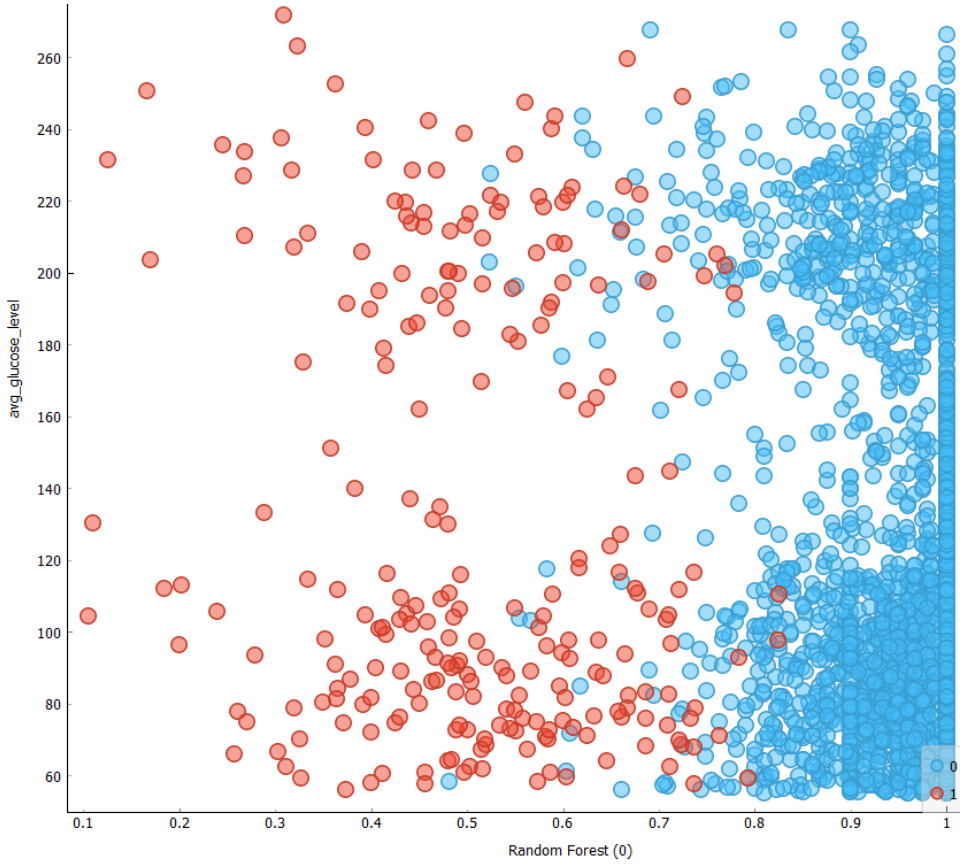

Figura 38 - Scatterplot Random Forest (secção3)

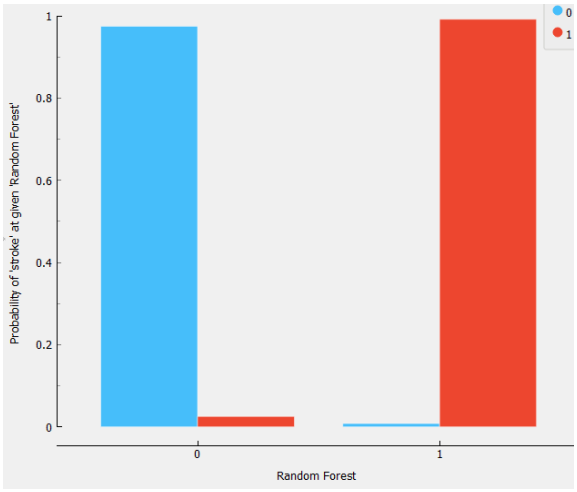

Figura 39 - Distribution Random Forest (secção3)

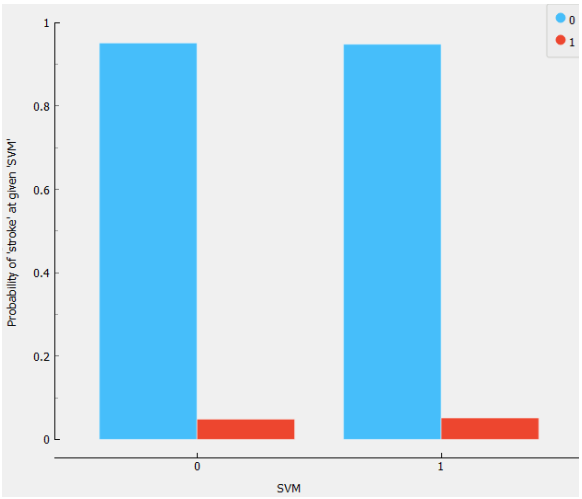

Figura 40 - Distribution SVM (secção3)

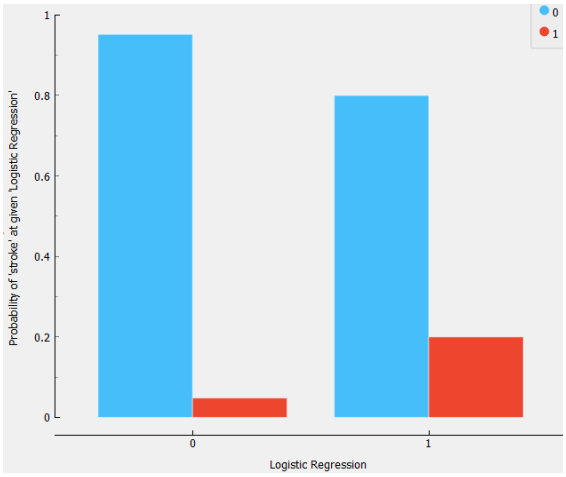

Figura 41 - Distribution Logistic Regression (secção3)

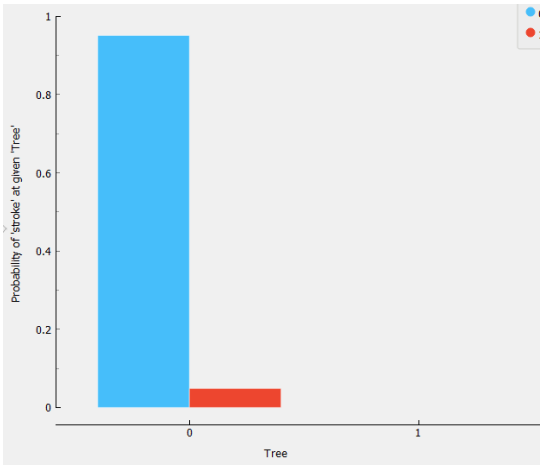

Figura 42 - Distribution Tree (secção3)

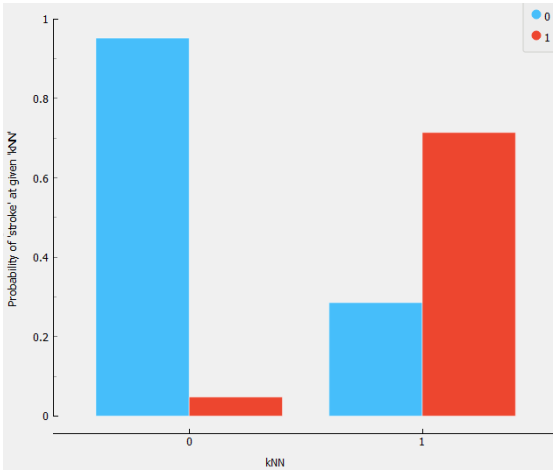

Figura 43 - Distribution KNN (secção3)

Como podemos observar os algoritmos com melhor precisão foram o KNN e o Random Forest.

Agora vou passar a fazer Test and score com cross validation:

Model	AUC	CA	F1	Precision	Recall
kNN	0.525	0.950	0.927	0.911	0.950
Tree	0.498	0.951	0.928	0.905	0.951
SVM	0.576	0.891	0.900	0.909	0.891
Random Forest	0.709	0.948	0.927	0.910	0.948
Logistic Regression	0.655	0.950	0.928	0.914	0.950

Figura 44 - Test and score Crossvalidation (secção3)

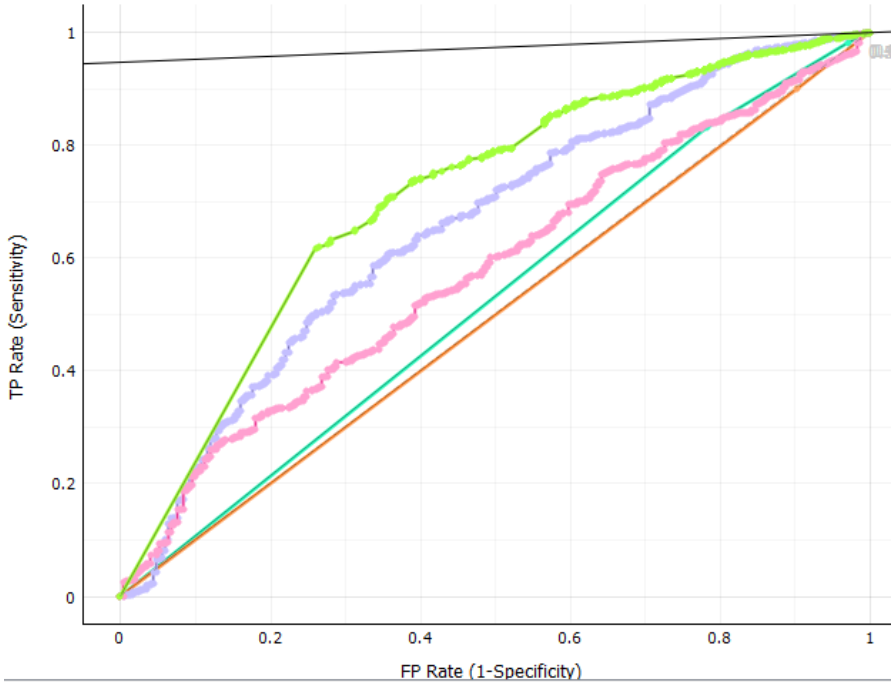

Figura 45 - ROC Crossvalidation (secção3)

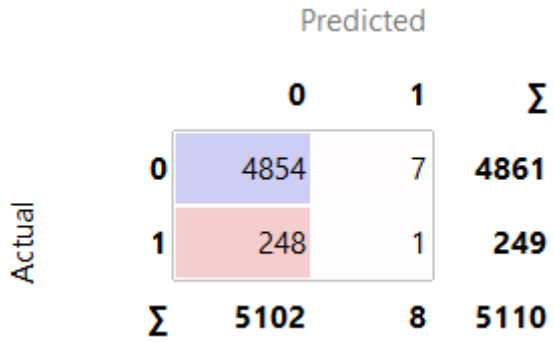

Figura 46 - Confusion Matrix KNN (secção3)

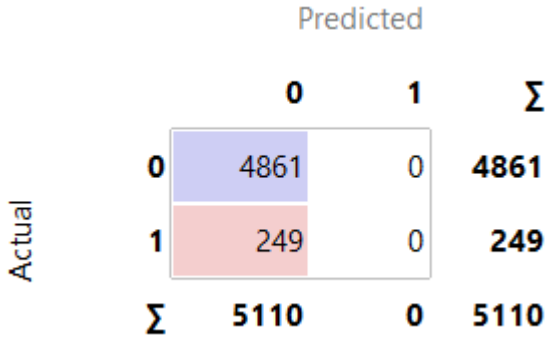

Figura 47 - Confusion Matrix Tree (secção3)

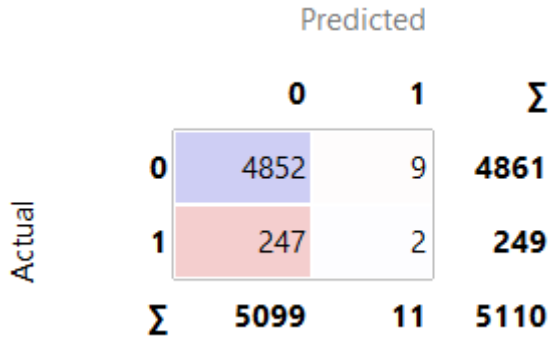

Figura 48 - Confusion Matrix Logistic Regression (secção3)

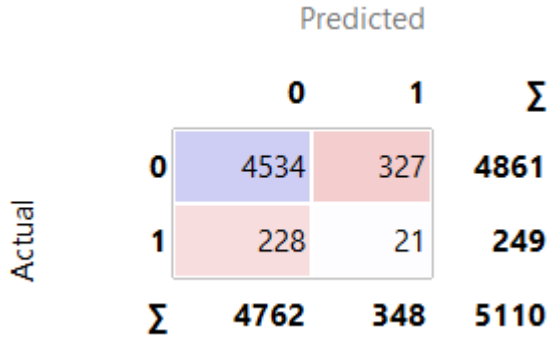

Figura 49 - Confusion Matrix SVM (secção3)

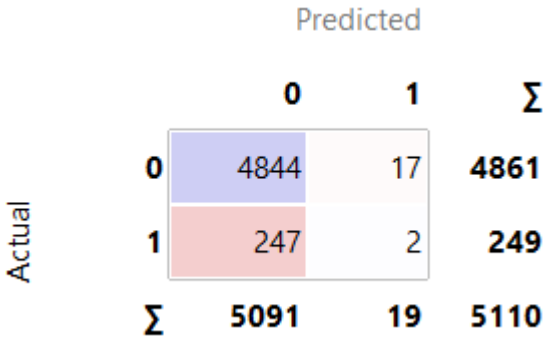

Figura 50 - Confusion Matrix Random Forest (secção3)

Nenhum algoritmo conseguiu prever mais que 20% dos casos com enfartes. Este resultado pode ser devido à dificuldade em adivinhar casos mais prováveis de enfarte e também à pouca informação para chegar a alguma conclusão.

A seguir usei o widget Rank para obter as features mais importantes:

	#	Gain ratio	Gini	ANOVA	χ^2
C heart disease	2	0.028	0.002	NA	87.987
N age		0.023	0.006	NA	232.273
C hypertension	2	0.019	0.002	NA	75.449
C ever married	2	0.011	0.001	NA	20.623
C work type	5	0.006	0.001	NA	2.926

Figura 51 - Rank (secção3)

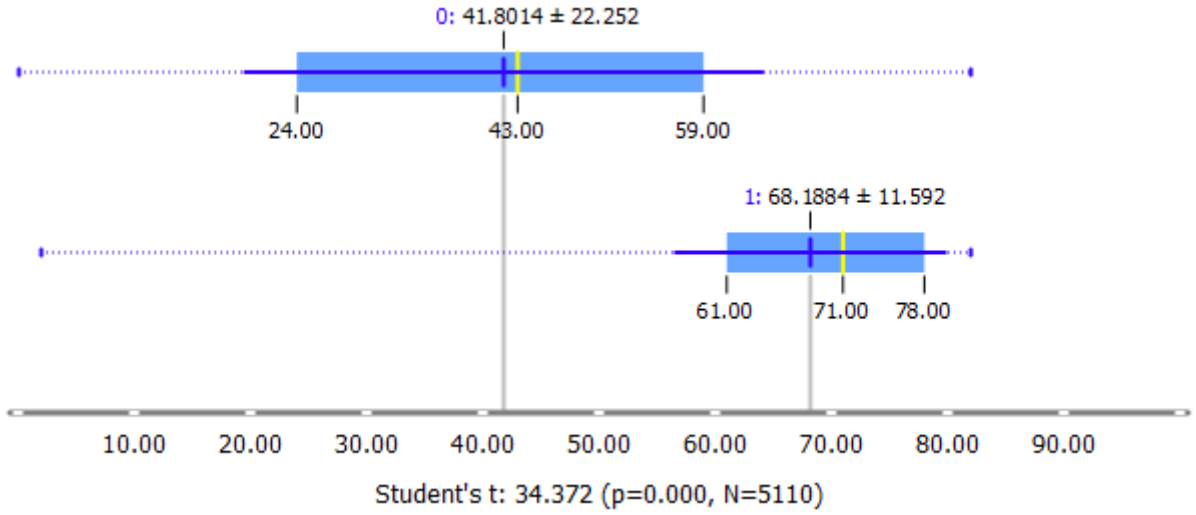

Figura 52 - Boxplot (secção3)

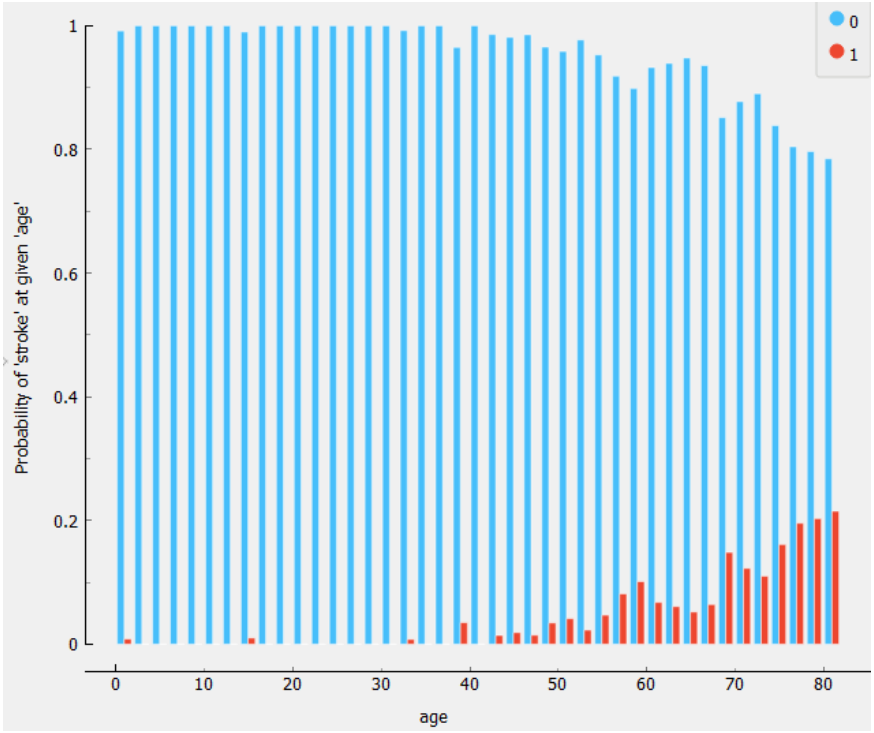

Figura 53 - Distribution Rank (secção3)

Como podemos observar tanto ter uma doença de coração como ser idoso aumenta bastante a probabilidade de ter um enfarte e, segundo o boxplot, a probabilidade de ter um enfarte com ambos é bastante alta.

Ao selecionar apenas as features que estiveram top 5 e recorrendo ao score Silhouette para definir o K do K-means observei:

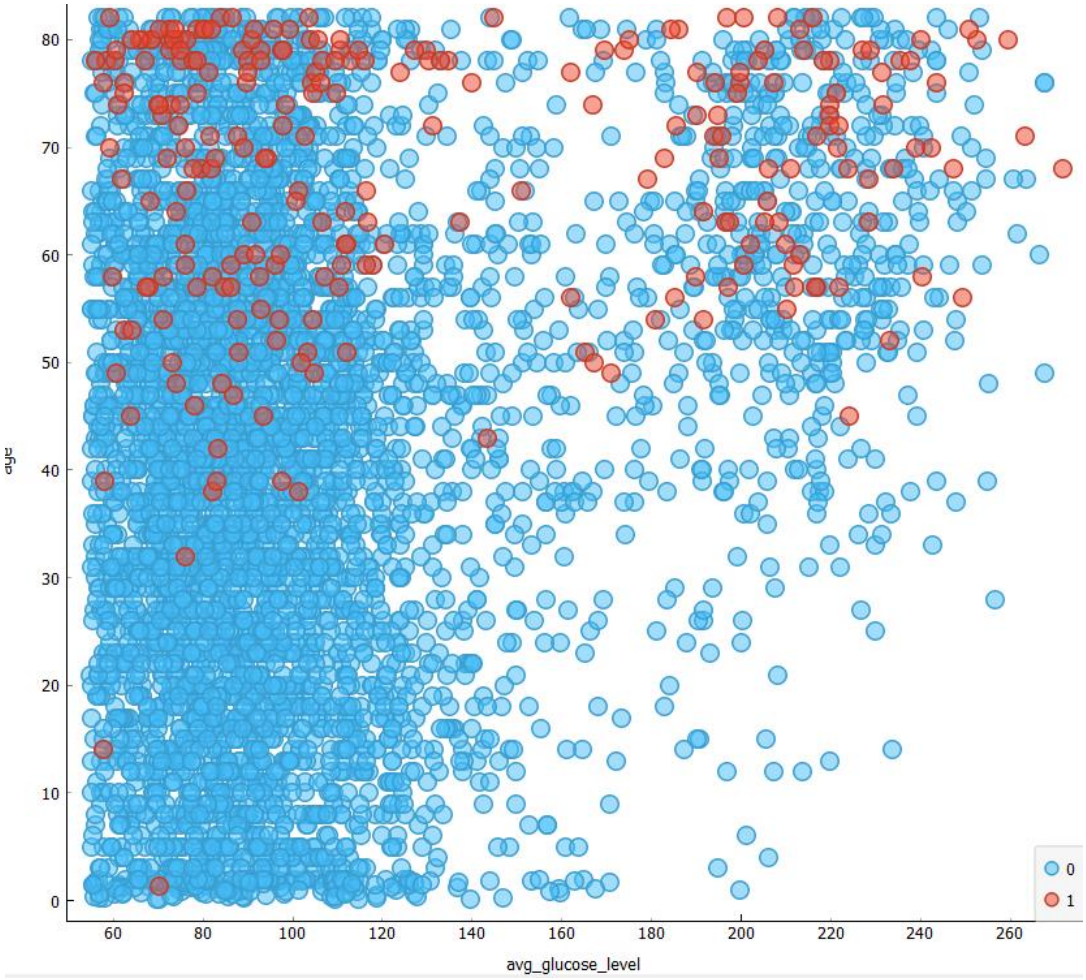

Figura 54 - Scatterplot kmeans (secção3)

Podemos observar que quanto maior é a idade maior é o average glucose level e probabilidade de enfarte.

Agora recorri ao widget distance para usar o algoritmo hierichical clustering e observei no scatterplot:

Figura 55 - Scatterplot HC (secção3)

Podemos observar que através do hierichical clustering este indica que apesar do average glucose level a idade de um humano é o que indica mais se este irá ter um infarte ou não.

5. Secção IV – Machine Learning (Python)

Nesta secção comecei primeiro por transformar o ficheiro fornecido .csv numa variável para manipular os dados do dataset stroke.csv.

Depois comecei por fazer uma análise das bibliotecas necessárias para suportar o código da secção 4 (matplotlib, numpy, pandas, sklearn, itertools).

A seguir comecei por fazer um menu onde contém os 4 algoritmos (KNN, DecisionTree, LogisticRegression, SVM):

```
df = pd.read_csv('stroke.csv')

while True:
    print("\nMenu:")
    print("0. Sair")

    print("Algoritmos:")
    print("1. KNN")
    print("2. Decision Trees")
    print("3. Logistic Regression")
    print("4. SVM")

    opcao = input("\nOpção: ")
```

Figura 56 - Menu secção 4

O código foi feito em Pycharm e JupiterNotebook e como tal irei usar ambos os “IDE’s” para apresentar resultados.

5.1. KNN

K-Nearest Neighbors é um algoritmo para aprendizado supervisionado. Onde os dados são 'treinados' com pontos de dados correspondentes à sua classificação. Uma vez que um ponto deve ser previsto, ele leva em consideração os pontos 'K' mais próximos a ele para determinar sua classificação.

```
print("Head:")
print(df.head())
input("")

print("Quantia de strokes:")
print(df['stroke'].value_counts())
input("")

print("Quantia de strokes em gráfico:")
df.hist(column='stroke', bins=50)
plt.show()
input("")

print("Features:")
print(df.columns)
print(df.dtypes)
input("")

print("Normalizar a data:")
varint = ['avg_glucose_level', 'bmi']
df[varint] = df[varint].astype("float")
varfloat = ['id', 'age', 'hypertension', 'heart_disease']
df[varfloat] = df[varfloat].astype("int")
avg_norm_loss = df['bmi'].astype("float").mean(axis=0)
df['bmi'].replace(np.nan, avg_norm_loss, inplace=True)
input("")
```

Figura 57 – KNN (1)

```
print("Converter a data frame do Pandas para vetor Numpy:")
X = df[['age', 'hypertension', 'heart_disease', 'avg_glucose_level', 'bmi', 'stroke']].values
print(X[0:5])
y = df['stroke'].values
print(y[0:5])
input("")

print("Train Test Split:")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
print('Train set:', X_train.shape, y_train.shape)
print('Test set:', X_test.shape, y_test.shape)
input("")

print("Training (K=4):")
k = 4
neigh = KNeighborsClassifier(n_neighbors=k)
neigh.fit(X_train, y_train)
input("")

print("Usar o modelo para prever o teste:")
yhat = neigh.predict(X_test)
print(yhat[0:5])
input("")

print("Accuracy evaluation:")
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
input("")
```

Figura 58 – KNN (2)

Primeiro comecei por visualizar os 5 primeiros resultados e reparei que o bmi tem valores NaN e então comecei por substituir todos os NaN pelo average dessa coluna.

Depois desenhei um gráfico que indica o número de pessoas que tiveram um stroke:

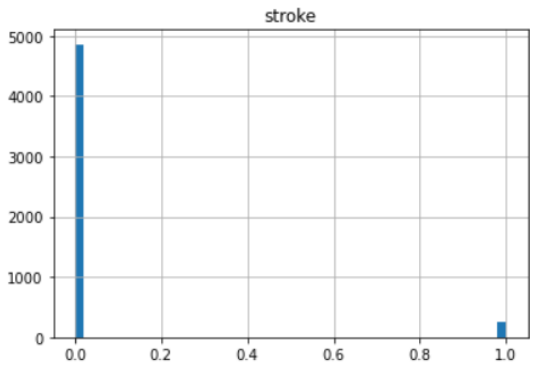

Figura 59 - Histograma Nº Strokes

Em segundo transformei certas variáveis em float pois o KNN terá melhor precisão se os números tiverem melhor precisão.

Em terceiro defini as labels e transformei o database do pandas para um array do numpy.

Em quarto normalizei o array.

Em quinto preparei o conteúdo do array fiz o train test split que consiste em transformar o dataset que tínhamos em duas variáveis train e test.

Em sexto definimos o k=4 e treinamos (fit) o modelo com a variável train.

Em sétimo usei a variável test para prever o modelo.

Em oitavo usei o train accuracy para ver se o train era acurated em relação ao test.

Por fim, usei uma função para calcular o valor de K para sabermos qual o melhor a usar para um determinado dataset:

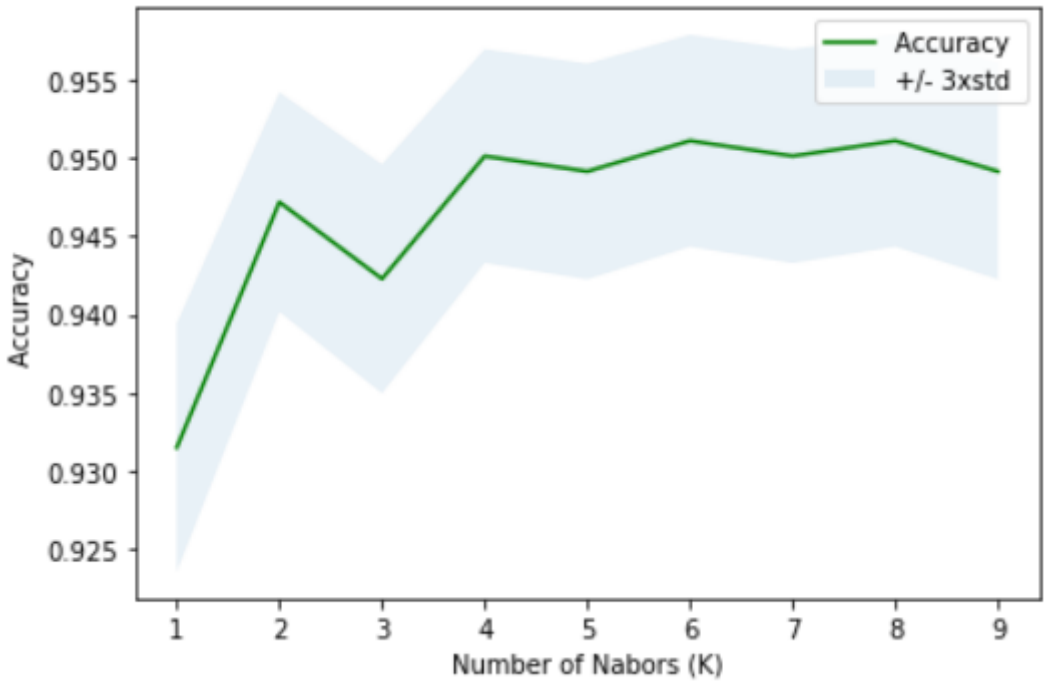

Figura 60 - Plotmodel accuracy K

5.2. Decision Tree

Uma árvore de decisão é uma ferramenta de suporte à decisão que usa um modelo de decisões em árvore e suas possíveis consequências, incluindo resultados de eventos casuais, custos de recursos e utilidade.

Comecei por formatar e normalizar o dataset comecei por definir os dummies:

```
print("Transformar variáveis categóricas em dummies:")
sex = preprocessing.LabelEncoder()
sex.fit(['Male', 'Female', 'Other'])
X[:, 0] = sex.transform(X[:, 0])
married = preprocessing.LabelEncoder()
married.fit(['Yes', 'No'])
X[:, 3] = married.transform(X[:, 3])
work = preprocessing.LabelEncoder()
work.fit(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_worked'])
X[:, 4] = work.transform(X[:, 4])
residence = preprocessing.LabelEncoder()
residence.fit(['Urban', 'Rural'])
X[:, 5] = residence.transform(X[:, 5])
smoke = preprocessing.LabelEncoder()
smoke.fit(['formerly smoked', 'never smoked', 'smokes', 'Unknown'])
X[:, 9] = smoke.transform(X[:, 9])
print(X[0:5])
```

Figura 61 - Dummies

Em primeiro comecei por fazer train test split.

Em segundo modelei a decision tree e defini como critéria entropy:

```
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
```

Figura 62 - Entropy

Em terceiro dei train ao dataset.

Em quarto dei predict.

E por fim avaliei a accuracy do decision tree, sendo que deu um valor perfeito (1.0):

```
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predictTree))
```

```
DecisionTrees's Accuracy:  1.0
```

Figura 63 - Decision Tree accuracy

5.3. Logistic Regression

O modelo logístico é usado para modelar a probabilidade de uma determinada classe ou evento existir como passa/falha, ganha/perde, vivo/morto, saudável/doente. Isso pode ser estendido para modelar várias classes de eventos, como determinar se uma imagem contém um gato, cachorro, leão, etc. Cada objeto detectado na imagem receberia uma probabilidade entre 0 e 1, com uma soma de um.

Comecei por formatar o dataset e normalizá-lo como fiz nos algoritmos anteriores.

Em primeiro comecei por fazer o train test split.

Em segundo defini como funciona a logistic regression:

```
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

```
LogisticRegression(C=0.01, solver='liblinear')
```

Figura 64 - Logistic Regression

Em terceiro dei predict ao dataset.

Em quarto para avaliar usei as métricas: Jaccard, Confusion matrix, F1 e LogLoss.

Jaccard:

```
jaccard_score(y_test, yhat)
```

```
0.0
```

Figura 65 - jaccard score

Confusion Matrix:

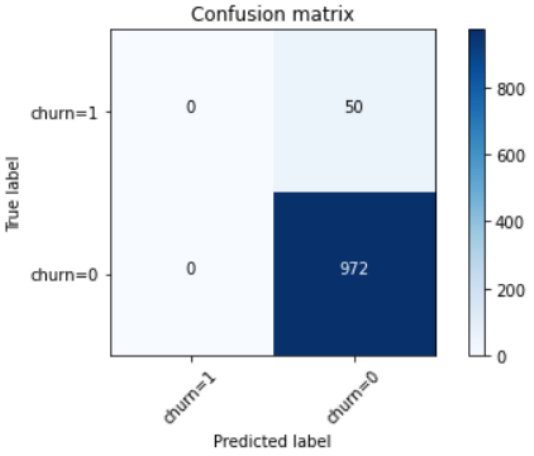

Figura 66 - confusion matrix

F1:

	precision	recall	f1-score	support
0	0.95	1.00	0.97	972
1	0.00	0.00	0.00	50
accuracy			0.95	1022
macro avg	0.48	0.50	0.49	1022
weighted avg	0.90	0.95	0.93	1022

Figura 67 - f1

LogLoss:

```
log_loss(y_test, yhat_prob)
```

```
0.19278357752741815
```

Figura 68 - logloss

5.4. SVM

O SVM toma como entrada um conjunto de dados e prediz, para cada entrada de dados, qual de duas classes possíveis a parte faz, o que faz do SVM um conjunto de entrada linear binário não probabilístico. Um modelo SVM é uma representação de exemplos como pontos no espaço, mapeados de maneira que os exemplos de cada categoria sejam tão amplos por um espaço tão amplo quanto possível.

Comecei por formatar o dataset e normalizá-lo como fiz nos algoritmos anteriores.

Em primeiro comecei por fazer um gráfico entre o bmi e o avg_glucose_level com relação ao stroke:

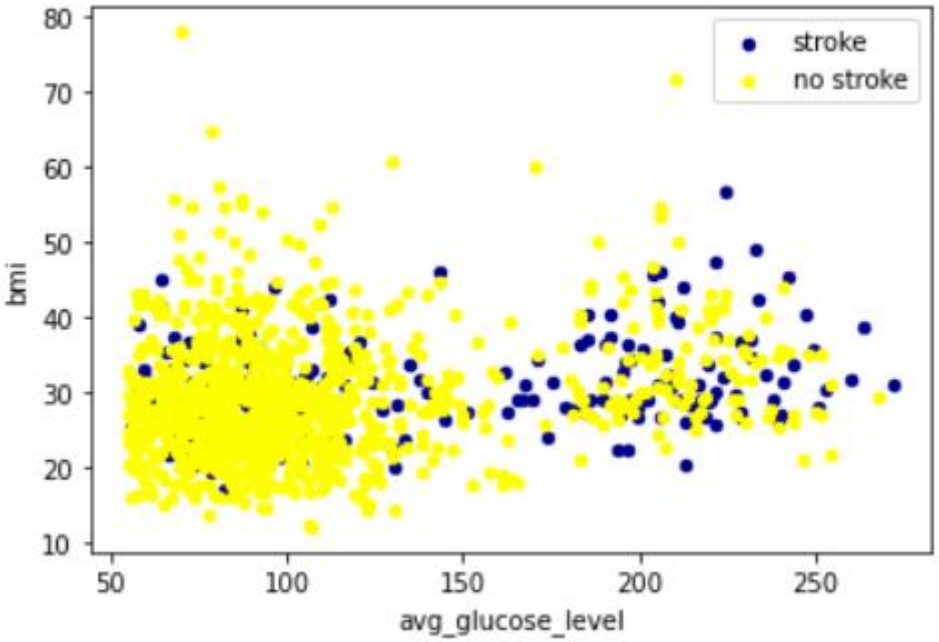

Figura 69 - bmi x avg_glucose_level - stroke

Em segundo lugar dei train test split.

Em terceiro defini o modelo de SVM e dei train:

```
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
```

```
SVC()
```

Figura 70 - SVM modelo

Em quarto dei predict ao modelo.

Em quinto usei as métricas: Confusion Matrix, f1 e jaccard:

Confusion Matrix:

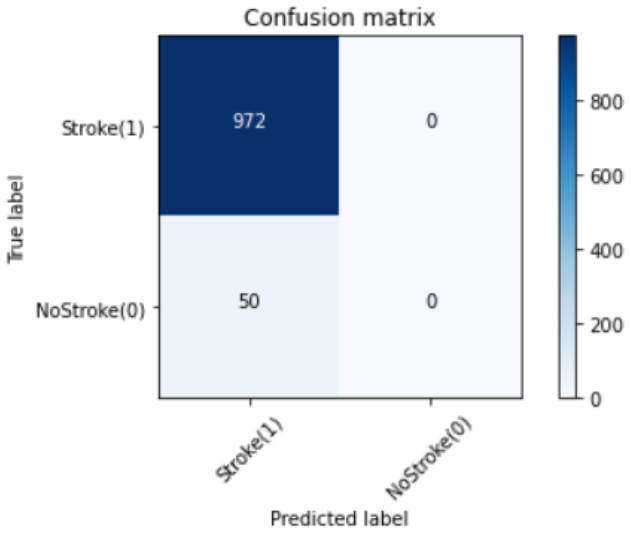

Figura 71 - confusion matrix SVM

F1:

```
f1_score(y_test, yhat, average='weighted')
```

```
0.9272278675557003
```

Figura 72 - f1 SVM

Jaccard:

```
jaccard_score(y_test, yhat)
```

```
0.0
```

Figura 73 - jaccard SVM

6. Análise SWOT

Gostei de realizar este relatório, pois ensinou-me a trabalhar com o Python para Machine Learning e aprofundou os meus conhecimentos de Orange Data Mining.

6.1. Strength

Devido ao estudo prévio da matéria teórica (e sendo que certas funções de machine learning saíram no teste), permitiu-me uma melhor compreensão e implementação da parte prática do trabalho.

6.2. Weaknesses

Algumas explicações podiam ser mais extensivas e aprofundadas.

6.3. Opportunities

Disponibilizar sempre fichas de trabalho (como esta) a IA antes de uma avaliação teórica pois ajuda o aluno na sua performance durante o exame/teste.

6.4. Threats

O dataset que o professor disponibilizou é demasiado grande e correu o risco de certos alunos não terem computadores com boa performance e assim não conseguirem fazer a secção 2.

7. Referências

- L. Veloso, “DataExploration.pdf”. ESTGOH: Engenharia informática.

- Kaggle Inc, “https://www.kaggle.com”.