UNIVERSIDADE ESTADUAL DE MONTES CLAROS
Centro de Ciências Exatas e Tecnológicas
Curso de Engenharia de Sistemas

Lucas Franklin Silva
Mateus Fellipe Alves Lopes

SISTEMAS ESPECIALISTAS
MINERAÇÃO DE DADOS

Montes Claros - MG
Outubro / 2018

Mineração de Dados

1. Introdução

1.1. Objetivos

Este trabalho tem como objetivo a implementação de algoritmos de *machine learning* para classificação e predição de dados reais. Utilizando os conceitos apresentados em sala de aula de pré processamento, treinamento e teste dos dados, além de avaliarmos os resultados obtidos para geração do conhecimento.

A turma foi dividida em duplas com o desafio de que cada dupla escolhesse dois datasets reais, disponíveis na UCI Machine Learning Repository . Um dataset deveria ser um problema de classificação enquanto que o outro um problema de regressão, para cada dataset devem ser utilizado alguns algoritmos, em Python, e suas acurácias devem ser comparadas. Para o problema de classificação optamos por escolher um dataset sobre readmissão de pacientes com diabetes em hospitais, disponível na UCI como Facebook Comment Volume Dataset Data Set. Enquanto que para o problema de regressão escolhemos um dataset sobre volume de comentários em uma postagem no Facebook, disponível na UCI como Diabetes 130-US hospitals for years 1999-2008 Data Set.

1.2. Diabetes 130-US hospitals for years 1999-2008

Esse dataset contém informações de readmissão de pacientes com diabetes em 130 hospitais americanos em um período de 9 anos, de 1999 até 2018.

Uma readmissão hospitalar é quando um paciente que recebe alta do hospital é readmitido novamente dentro de um determinado período de tempo. As taxas de readmissão hospitalar para determinadas condições são agora consideradas um indicador de qualidade hospitalar e também afetam adversamente o custo do tratamento.

Algumas informações sobre Data Set:

Número Total de Instâncias	Número Total de Atributos	Variáveis Contínuas	Variáveis de Categorias	Dados Faltando
101.766	55	8	6	Sim

1.3. **Facebook Comment Volume Dataset Data Set**

As instâncias neste conjunto de dados contêm recursos extraídos de publicações no Facebook. A tarefa associada aos dados é prever quantos comentários a postagem receberá.

Atualmente a previsão de comentários pode ser uma ferramenta muito útil para usuários e páginas do Facebook, sendo possível prever o impacto de anúncios e propagandas, além de quantidade de pessoas alcançadas.

Algumas informações sobre Data Set:

Número Total de Instâncias	**Número Total de Atributos**	**Variáveis Contínuas**	**Variáveis de Categorias**	**Dados Faltando**
602403	54	-	14	Não

2. Metodologia

Para implementação dos métodos e dos algoritmos utilizamos o Python 3 no Jupyter, uma das principais plataformas utilizadas para manipulação e mineração de dados, além das principais bibliotecas de data science no python, como pandas, numpy e sklearn.

Primeiramente, analisamos cada conjunto de dados e verificamos se haviam dados faltantes e se todos os dados eram numéricos. A única base de dados que necessitava de um tratamento é a base de readmissão de pacientes com diabetes, assim inicialmente verificamos a quantidade de dados faltantes e optamos por remover tais atributos da base de dados, os dados removidos podem ser visualizados na tabela a seguir:

Atributo	**Quantidade de dados faltantes**
race	2273
weight	98569
payer_code	40256
medical_specialty	49949
diag_1	21
diag_2	358
diag_3	1423

Em seguida, transformamos valores não numéricos em numéricos e normalização todos dentro da mesma escala.

Após toda a base de dados ter sido pré processada escolhemos alguns algoritmos da literatura para avaliarmos seus desempenhos, para o dataset de readmissão de pacientes com diabetes, problema de classificação, utilizamos os classificadores Máquina de vetor de suporte (SVM), Bayesiano e Árvore de Decisão. Para o dataset de volume de comentários do Facebook, problema de regressão, utilizamos os métodos Bayesiano, Árvore de Decisão e Random Forest.

Para treinamento dos modelos testamos inicialmente a Validação Simples com 80% dos dados para treinamento e 20% para teste e depois a Validação Cruzada dividindo os dados em 10 grupos.

3. Resultados e Discussões

3.1. Diabetes 130-US hospitals for years 1999-2008

Comparando primeiramente os valores do dataset de readmissão de pacientes com diabetes obtivemos os seguintes valores para os classificadores, de acordo com o tipo de validação:

	Árvore de Decisão	**Bayesiano**	**SVM**
Validação Simples	61,61%	59,63%	60,52%
Cross Validation	60,23%	60,18%	59,28%

Podemos observar que todos os métodos obtiveram um desempenho próximo, nenhum se destacou dentre os demais classificadores.

Na tabela a seguir observarmos a quantidade mínima de features sem comprometer o desempenho do classificador.

Árvore de Decisão	**Bayesiano**	**SVM**
12	6	7

Na tabela a seguir veremos o comparação dos desempenhos dos classificadores de acordo com a quantidade de features, podemos observar que os algoritmos sofreram pioras devido ao aumento de features diminuindo a precisão do modelo.

	Árvore de Decisão	**Bayesiano**	**SVM**
Melhor Número de Features	60,23%	60,18%	59,28%
Todas as Features	56,45%	57,11%	53,74%

Sobre os parâmetros e valores, a próxima tabela contém a configuração dos melhores parâmetros para os melhores desempenhos dos métodos.

	Árvore de Decisão	**Bayesiano**	**SVM**
Parâmetros	criterion: entropy	GaussianNB	decision_function_shape: ovr
Valores	max_depth: 8	-	max_iter -1

As seguintes features foram consideradas as mais decisivas na construção do modelo:

Árvore de Decisão	**Bayesiano**	**SVM**
● Idade ● Tempo Hospital	● Número de visitas anteriores ● Tipo de admissão	● Número de visitas anteriores ● Serviços utilizados ● Acetohexamide

3.2. Facebook Comment Volume Dataset Data Set

Comparando os valores do dataset de volume de comentários em uma postagem no Facebook obtivemos os seguintes valores para os classificadores, de acordo com o tipo de validação:

	Árvore de Decisão	**Bayesiano**	**Random Forest**
Todas as Features	51%	30%	77%
Seis Melhores Features	16%	15%	61,36%

Podemos observar que a Random Forest obteve os melhores resultados entre os 3 métodos, com e sem todas as features.entre os demais classificadores.

Na tabela a seguir observarmos a quantidade mínima de features sem comprometer o desempenho do classificador. Diferente do problema anterior, na regressão necessitamos da maioria das features possíveis para obter bons resultados

Árvore de Decisão	**Bayesiano**	**Random Forest**
54	54	54

Sobre os parâmetros e valores, a tabela a seguir contém a configuração dos melhores parâmetros para os melhores desempenhos dos métodos.

	Árvore de Decisão	**Bayesiano**	**Random Forest**
Parâmetros	criterion: mse	default	criterion: mse
Valores	max_depth: 12	default	n_estimators: 125

4. Conclusão

Através dos modelos apresentados foi possível verificar o funcionamento dos algoritmos de classificação e regressão. Pelo fato da complexidade da base de dados de classificação, os algoritmos alcançaram uma acurácia relativamente baixa, mais maior que 60%. Alguns dos algoritmos de regressão alcançaram uma acurácia baixa, mas a floresta randômica apresentou um excelente resultado de 77% utilizando de todas as features da base de dados.

5. Anexos: Códigos

5.1 Classificação

5.1.1 Árvore de Decisão

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

 df = pd.read_csv("./diabetes_data_preprocessed.csv")

 X = df[['gender', 'race','discharge_disposition_id', 'time_in_hospital',
 'age','num_medications', 'num_procedures', 'admission_type_id',
 'service_utilization','numchange','nummed']].values
 y = df["readmitted"]

 from sklearn.model_selection import train_test_split

 X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3,
 random_state=3)

drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 8)

drugTree.fit(X_trainset,y_trainset)

 predTree = drugTree.predict(X_testset)

 from sklearn import metrics
 import matplotlib.pyplot as plt
 print("Precisão: {:.2%}".format(metrics.accuracy_score(y_testset, predTree)))
 print("Cross Validation: {:.2%}".format(np.mean(cross_val_score(predTree,
 X_trainset, y_trainset, cv=10))))
```

5.1.2 SVM

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.model_selection import cross_val_score

data = pd.read_csv("data/diabetes_data_preprocessed.csv")

X_train, X_test = train_test_split(data, test_size=0.2,
random_state=int(time.time()))
used_features = [
    'encounter_id',
     'patient_nbr', 'race', 'gender', 'age', 'admission_type_id',
     'discharge_disposition_id', 'admission_source_id', 'time_in_hospital',
     'num_lab_procedures', 'num_procedures', 'num_medications',
     'number_outpatient', 'number_emergency', 'number_inpatient', 'diag_1',
     'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult',
     'metformin', 'repaglinide', 'nateglinide',
     'glimepiride', 'acetohexamide', 'glipizide', 'tolbutamide',
     'pioglitazone',
     'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin',
      'change', 'diabetesMed', 'readmitted',
     'service_utilization', 'numchange', 'nummed',
]

clf = svm.SVC().fit(
    X_train[used_features].values,
    X_train["readmitted"]
)

print("Precisão: {:.2%}".format(clf.score( X_test[used_features].values,
X_test["readmitted"])))
```

5.1.3 Bayesino

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
from sklearn.model_selection import cross_val_score

data = pd.read_csv("data/diabetes_data_preprocessed.csv")

data.head()

data.head()

used_features = [
        'gender', 'age', 'admission_type_id',
      'discharge_disposition_id', 'admission_source_id', 'time_in_hospital',
      'num_lab_procedures', 'num_procedures', 'num_medications',
      'number_outpatient', 'number_emergency', 'number_inpatient', 'diag_1',
      'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult',
      'metformin',  'chlorpropamide',
      'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
      'pioglitazone', 'miglitol', 'troglitazone',
      'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin',
      'glimepiride-pioglitazone', 'metformin-rosiglitazone',
      'metformin-pioglitazone', 'change', 'diabetesMed',
      'service_utilization', 'level1_diag1',
      'level2_diag1', 'level1_diag2', 'level2_diag2', 'level1_diag3',
      'level2_diag3'
]

X_train, X_test = train_test_split(data, test_size=0.2,
random_state=int(time.time()))
gnd = GaussianNB()

gnd.fit(
    X_train[used_features].values,
    X_train["readmitted"]
)
y_pred = gnd.predict(X_test[used_features])
```

```
print("Precisão: {:.2%}".format(gnd.score( X_test[used_features].values,
X_test["readmitted"])))
print("Cross Validation: {:.2%}".format(np.mean(cross_val_score(gnd,
X_test[used_features].values, X_test["readmitted"], cv=10))))
```

5.2 Regressão

5.2.1 Árvore de Decisão

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

dataset = pd.read_csv("data_facebook.csv", low_memory=False)
dataset.drop(dataset.index[0], inplace=True)
dataset.drop(dataset.index[0], inplace=True)

X = dataset[['Feature 11','Feature 26','Feature 16', 'Feature 36', 'Feature 35','Feature
30']].values
y = dataset['Feature 54']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)

from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
y_pred)))

metrics.r2_score(y_test, y_pred)
```

5.2.2 Bayesiano

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import BayesianRidge
from sklearn import metrics

data = pd.read_csv("Features_Variant_1.csv")

used_features = [
    'Feature 1', 'Feature 2', 'Feature 3', 'Feature 4', 'Feature 5',
     'Feature 6', 'Feature 7', 'Feature 8', 'Feature 9', 'Feature 10',
     'Feature 11', 'Feature 12', 'Feature 13', 'Feature 14', 'Feature 15',
     'Feature 16', 'Feature 17', 'Feature 18', 'Feature 19', 'Feature 20',
     'Feature 21', 'Feature 22', 'Feature 23', 'Feature 24', 'Feature 25',
     'Feature 26', 'Feature 27', 'Feature 28', 'Feature 29', 'Feature 30',
     'Feature 31', 'Feature 32', 'Feature 33', 'Feature 34', 'Feature 35',
     'Feature 36', 'Feature 37', 'Feature 38', 'Feature 39', 'Feature 40',
     'Feature 41', 'Feature 42', 'Feature 43', 'Feature 44', 'Feature 45',
     'Feature 46', 'Feature 47', 'Feature 48', 'Feature 49', 'Feature 50',
     'Feature 51', 'Feature 52', 'Feature 53'
]

X_train, X_test = train_test_split(data, test_size=0.3, random_state=int(time.time()))
clf = BayesianRidge()
clf.fit(
   X_train[used_features].values,
   X_train["Feature 54"]
)
y_pred = clf.predict(X_test[used_features])

print("Precisão: {:.2%}".format(clf.score( X_test[used_features].values, X_test["Feature
54"])))
print(metrics.r2_score(X_test["Feature 54"],y_pred))
print("Cross Validation: {:.2%}".format(np.mean(cross_val_score(clf,
X_test[used_features].values, X_test["Feature 54"], cv=10))))
```

5.2.2 Floresta Randômica

```
import numpy as np
import pandas as pd

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from subprocess import check_output
from datetime import time

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

model = RandomForestRegressor(n_jobs=-1)

estimators = np.arange(10, 200, 10)
scores = []
for n in estimators:
    model.set_params(n_estimators=n)
    model.fit(X_train, y_train)
    scores.append(model.score(X_test, y_test))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)

np.mean(scores)
```