Introdução ao Aprendizado de Máquina

Relatório do Trabalho Prático 2 - EEL891 - Semestre 2023/2

Orientador: Heraldo Luis -- Aluno: Lucas Tavares Da Silva Ferreira

This document is the report of the 2nd practical task of the discipline "Introduction to Machine Learning" (EEL891) at the Federal University of Rio de Janeiro (UFRJ). The work consists of building a classifier to support the credit approval decision.

In this work, multivariable regression techniques will be used to estimate the price of a property based on characteristics such as the type of property (apartment, house, loft or studio apartment), neighborhood where it is located, number of rooms, number of spaces, useful area , extra area and presence of differentiating elements in relation to other properties, such as barbecue, parking for visitors, swimming pool, playground, sports court, football field, party room, games room, gym, sauna and views of the sea.

Trabalho 2

Tratamento de variáveis

In [ ]:

```
import pandas as pd

import math
from scipy.stats import pearsonr
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

#-------------------------------------------------------------------------------
#  Importando os dados e os aplicando em um dataframe
#-------------------------------------------------------------------------------

dados= pd.read_csv('conjunto_de_treinamento.csv', delimiter=',', decimal='.')
dados_teste = pd.read_csv('conjunto_de_teste.csv', delimiter=',', decimal='.')
```

In [ ]:

```
dados = dados.drop(['Id'], axis = 1)
dados_teste = dados_teste.drop(['Id'], axis = 1)
```

Subcategorização dos dados

In [ ]:

```
diferenciais = {'campo de futebol e copa':'futebol+',
 'campo de futebol e esquina':'futebol+',
 'campo de futebol e estacionamento visitantes':'futebol+',
```

```
'campo de futebol e playground':'futebol+',
'campo de futebol e quadra poliesportiva':'futebol+',
'campo de futebol e salao de festas':'futebol+',
'children care':'children care',
'children care e playground':'children care+',
'churrasqueira':'churrasco',
'churrasqueira e campo de futebol':'churrasco+',
'churrasqueira e copa':'churrasco+',
'churrasqueira e esquina':'churrasco+',
'churrasqueira e estacionamento visitantes':'churrasco+',
'churrasqueira e frente para o mar':'churrasco+',
'churrasqueira e playground':'churrasco+',
'churrasqueira e sala de ginastica':'churrasco+',
'churrasqueira e salao de festas':'churrasco+',
'churrasqueira e sauna':'churrasco+',
'copa':'copa',
'copa e esquina':'copa+',
'copa e estacionamento visitantes':'copa+',
'copa e playground':'copa+',
'copa e quadra poliesportiva':'copa+',
'copa e sala de ginastica':'copa+',
'copa e salao de festas':'copa+',
'esquina':'esquina',
'esquina e estacionamento visitantes':'esquina+',
'esquina e playground':'esquina+',
'esquina e quadra poliesportiva':'esquina+',
'esquina e sala de ginastica':'esquina+',
'esquina e salao de festas':'esquina+',
'estacionamento visitantes':'estacionamento visitantes',
'estacionamento visitantes e playground':'estacionamento visitantes+',
'estacionamento visitantes e sala de ginastica':'estacionamento visitantes+',
'estacionamento visitantes e salao de festas':'estacionamento visitantes+',
'frente para o mar':'frente para o mar',
'frente para o mar e campo de futebol':'mar+',
'frente para o mar e copa':'mar+',
'frente para o mar e esquina':'mar+',
'frente para o mar e playground':'mar+',
'frente para o mar e quadra poliesportiva':'mar+',
'frente para o mar e salao de festas':'mar+',
'nenhum':'nenhum',
'piscina':'piscina',
'piscina e campo de futebol':'piscina+',
'piscina e children care':'piscina+',
'piscina e churrasqueira':'piscina+',
'piscina e copa':'piscina+',
'piscina e esquina':'piscina+',
'piscina e estacionamento visitantes':'piscina+',
'piscina e frente para o mar':'piscina+',
'piscina e hidromassagem':'piscina+',
'piscina e playground':'piscina+',
'piscina e quadra de squash':'piscina+',
'piscina e quadra poliesportiva':'piscina+',
'piscina e sala de ginastica':'piscina+',
'piscina e salao de festas':'piscina+',
'piscina e salao de jogos':'piscina+',
'piscina e sauna':'piscina+',
'playground':'playground',
'playground e quadra poliesportiva':'playground+',
'playground e sala de ginastica':'playground+',
'playground e salao de festas':'playground+',
'playground e salao de jogos':'playground+',
'quadra poliesportiva':'quadra poliesportiva',
'quadra poliesportiva e salao de festas':'quadra+',
'sala de ginastica':'sala de ginastica',
'sala de ginastica e salao de festas':'ginastica+',
'sala de ginastica e salao de jogos':'ginastica+',
'salao de festas':'salao de festas',
'salao de festas e salao de jogos':'festa+',
'salao de festas e vestiario':'festa+',
'salao de jogos':'salao de jogos',
'sauna':'sauna',
'sauna e campo de futebol':'sauna+',
```

```
 'sauna e copa':'sauna+',
 'sauna e esquina':'sauna+',
 'sauna e frente para o mar':'sauna+',
 'sauna e playground':'sauna+',
 'sauna e quadra poliesportiva':'sauna+',
 'sauna e sala de ginastica':'sauna+',
 'sauna e salao de festas':'sauna+',
 'vestiario':'vestiario'}


dados = dados.replace(diferenciais)
dados_teste = dados_teste.replace(diferenciais)
```

In [ ]:

```
bairros = {'Aflitos': 'pobre',
 'Afogados':'pobre',
 'Agua Fria':'pobre',
 'Apipucos':'meiotermo',
 'Areias':'meiotermo',
 'Arruda':'meiotermo',
 'Barro':'pobre',
 'Beberibe':'pobre',
 'Beira Rio':'pobre',
 'Benfica':'pobre',
 'Boa Viagem':'rico',
 'Boa Vista':'meiotermo',
 'Bongi':'meiotermo',
 'Cajueiro':'meiotermo',
 'Campo Grande':'meiotermo',
 'Casa Amarela':'meiotermo',
 'Casa Forte':'rico',
 'Caxanga':'meiotermo',
 'Centro':'meiotermo',
 'Cid Universitaria':'meiotermo',
 'Coelhos':'pobre',
 'Cohab':'pobre',
 'Cordeiro':'meiotermo',
 'Derby':'meiotermo',
 'Dois Irmaos':'meiotermo',
 'Encruzilhada':'meiotermo',
 'Engenho do Meio':'meiotermo',
 'Espinheiro':'rico',
 'Estancia':'pobre',
 'Fundao':'pobre',
 'Gracas':'rico',
 'Guabiraba':'pobre',
 'Hipodromo':'pobre',
 'Ilha do Leite':'meiotermo',
 'Ilha do Retiro':'rico',
 'Imbiribeira':'meiotermo',
 'Ipsep':'meiotermo',
 'Iputinga':'meiotermo',
 'Jaqueira':'rico',
 'Jd S Paulo':'meiotermo',
 'Lagoa do Araca':'meiotermo',
 'Macaxeira':'pobre',
 'Madalena':'rico',
 'Monteiro':'meiotermo',
 'Paissandu':'rico',
 'Parnamirim':'meiotermo',
 'Piedade':'pobre',
 'Pina':'meiotermo',
 'Poco':'pobre',
 'Poco da Panela':'meiotermo',
 'Ponto de Parada':'pobre',
 'Prado':'meiotermo',
 'Recife':'meiotermo',
 'Rosarinho':'meiotermo',
 'S Jose':'rico',
 'San Martin':'meiotermo',
```

```
 'Sancho':'meiotermo',
 'Santana':'meiotermo',
 'Setubal':'rico',
 'Soledade':'meiotermo',
 'Sto Amaro':'meiotermo',
 'Sto Antonio':'meiotermo',
 'Tamarineira':'rico',
 'Tejipio':'meiotermo',
 'Torre':'meiotermo',
 'Torreao':'meiotermo',
 'Varzea':'meiotermo',
 'Zumbi':'pobre'}


dados = dados.replace(bairros)
dados_teste = dados_teste.replace(bairros)
```

Aplicando One-Hot-Encoding

In [ ]:

```
dados = pd.get_dummies(dados,columns = ['tipo'])
dados = pd.get_dummies(dados,columns = ['bairro'])
dados = pd.get_dummies(dados,columns = ['tipo_vendedor'])
dados = pd.get_dummies(dados,columns = ['diferenciais'])

dados_teste  = pd.get_dummies(dados_teste ,columns = ['tipo'])
dados_teste  = pd.get_dummies(dados_teste ,columns = ['bairro'])
dados_teste  = pd.get_dummies(dados_teste ,columns = ['tipo_vendedor'])
dados_teste  = pd.get_dummies(dados_teste ,columns = ['diferenciais'])



#display(dados)
```

In [ ]:

```
print(dados.columns.tolist())
```

In [ ]:

```
print(dados_teste.columns.tolist())
```

Retirar colunas não presentes em ambos arquivos

In [ ]:

```
colunasDados = dados.columns.tolist()
colunasDadosTeste = dados_teste.columns.tolist()
e = (set(colunasDados) & set(colunasDadosTeste))
print("Elementos comuns", e )
```

In [ ]:

```
colunasDados = dados.columns.tolist()
colunasDadosTeste = dados_teste.columns.tolist()
a = set(colunasDadosTeste)
b = set(colunasDados)
f = a-e
g = b-e
print(f)
print(g)
```

In [ ]:

```
dados_teste = dados_teste.drop(['diferenciais_frente para o mar e children care', 'bairr
o_Ibura', 'diferenciais_estacionamento visitantes e hidromassagem', 'diferenciais_campo d
```

```
e futebol e sala de ginastica', 'diferenciais_estacionamento visitantes e salao de jogos'
, 'diferenciais_frente para o mar e hidromassagem', 'diferenciais_hidromassagem e salao
de festas', 'diferenciais_churrasqueira e children care', 'diferenciais_copa e hidromassa
gem'],axis = 1)
dados = dados.drop(['tipo_Quitinete', 'diferenciais_children care', 'diferenciais_childr
en care+', 'diferenciais_quadra poliesportiva'],axis = 1)
```

Retirando os outliers

In [ ]:

```
dados = dados[(dados['preco']>= 50000) & (dados['preco']<= 5000000)]
```

Verificando o coeficiente de Pearson

In [ ]:

```
variaveis_categoricas = [ x for x in dados.columns if dados[x].dtype == 'object']
for v in variaveis_categoricas:
  print('\n%15s: '%v, "%4d categorias" % len(dados[v].unique()))
  print(dados[v].unique(),'\n')
```

In [ ]:

```
colunasNovas = dados.columns
for col in colunasNovas:
  print('%10s = %6.3f' % (col, pearsonr(dados[col],dados['preco'])[0]))
```

Removendo variáveis de baixo coeficiente de Pearson

In [ ]:

```
dados = dados.drop(['quadra','s_ginastica','diferenciais_copa','diferenciais_esquina+','d
iferenciais_sauna','diferenciais_vestiario'],axis=1)
dados_teste = dados_teste.drop(['quadra','s_ginastica','diferenciais_copa','diferenciais_
esquina+','diferenciais_sauna','diferenciais_vestiario'],axis=1)

dados = dados.drop(['diferenciais_ginastica+','diferenciais_quadra+','diferenciais_salao
de jogos'],axis=1)
dados_teste = dados_teste.drop(['diferenciais_ginastica+','diferenciais_quadra+','diferen
ciais_salao de jogos'],axis=1)


dados = dados.drop(['diferenciais_festa+','diferenciais_piscina','diferenciais_churrasco'
,'bairro_pobre'],axis=1)
dados_teste = dados_teste.drop(['diferenciais_festa+','diferenciais_piscina','diferencia
is_churrasco','bairro_pobre'],axis=1)

dados = dados.drop(['s_jogos','tipo_vendedor_Imobiliaria','tipo_vendedor_Pessoa Fisica','
diferenciais_futebol+','diferenciais_frente para o mar'],axis=1)
dados_teste = dados_teste.drop(['s_jogos','tipo_vendedor_Imobiliaria','tipo_vendedor_Pess
oa Fisica','diferenciais_futebol+','diferenciais_frente para o mar'],axis=1)

dados = dados.drop(['diferenciais_estacionamento visitantes+','diferenciais_sauna+','dife
renciais_esquina'],axis=1)
dados_teste = dados_teste.drop(['diferenciais_estacionamento visitantes+','diferenciais_s
auna+','diferenciais_esquina'],axis=1)


dados = dados.drop(['tipo_Loft','diferenciais_churrasco+','diferenciais_playground','dife
renciais_playground+'],axis=1)
dados_teste = dados_teste.drop(['tipo_Loft','diferenciais_churrasco+','diferenciais_playg
round','diferenciais_playground+'],axis=1)

dados = dados.drop(['diferenciais_salao de festas','diferenciais_estacionamento visitante
s','estacionamento','area_extra','churrasqueira'],axis=1)
```

```
dados_teste = dados_teste.drop(['diferenciais_salao de festas','diferenciais_estacionamento visitantes','estacionamento','area_extra','churrasqueira'],axis=1)

dados = dados.drop(['playground','diferenciais_copa+'],axis=1)
dados_teste = dados_teste.drop(['playground','diferenciais_copa+'],axis=1)

dados = dados.drop(['s_festas','diferenciais_sala de ginastica'],axis=1)
dados_teste = dados_teste.drop(['s_festas','diferenciais_sala de ginastica'],axis=1)
```

Separação dos conjuntos de treino e teste

In [ ]:

```
x = dados.drop('preco',axis = 1).to_numpy()
y = dados['preco'].to_numpy()
```

In [ ]:

```
x_treino, x_teste, y_treino, y_teste = train_test_split(
        x,
        y,
        test_size = 1500,
        random_state = 0
)
```

Scaller para calibragem

In [ ]:

```
escala = StandardScaler()
escala.fit(x_treino)

x_treino = escala.transform(x_treino)
x_teste = escala.transform(x_teste)
```

In [ ]:

```
#scaler = MinMaxScaler()
#scaler.fit_transform(x_treino)
#x_treino = scaler.transform(x_treino)
#x_teste = scaler.transform(x_teste)
```

Linear Regression

In [ ]:

```
#treinar um regressor linear

regressor_linear = LinearRegression()

regressor_linear = regressor_linear.fit(x_treino,y_treino)
```

In [ ]:

```
y_resposta_treino = regressor_linear.predict(x_treino)
y_resposta_teste = regressor_linear.predict(x_teste)
```

In [ ]:

```
print(y_resposta_treino)
```

In [ ]:

```
mse_in = mean_squared_error(y_treino,y_resposta_treino)
rmse_in = math.sqrt(mse_in)
```

```
r2_in = r2_score(y_treino,y_resposta_treino)
rmspe = (np.sqrt(np.mean(np.square((y_teste - y_resposta_teste) / y_teste))))
mse_out = mean_squared_error(y_teste,y_resposta_teste)
rmse_out = math.sqrt(mse_out)
r2_out = r2_score(y_teste,y_resposta_teste)
print('%7s %17.4f %15.4f' % ( 'mse', mse_in, mse_out))
print('%7s %17.4f %15.4f' % ( 'rmse', rmse_in, rmse_out))
print('%7s %17.4f %15.4f' % ( 'r2', r2_in, r2_out))
rmspe = (np.sqrt(np.mean(np.square((y_teste - y_resposta_teste) / y_teste))))

print('%7s %17.4f'  % ( 'rmspe', rmspe))
```

Polinomial Regression

In [ ]:

```
for k in range (1,4):
  pf = PolynomialFeatures(degree = k)
  pf = pf.fit(x_treino)
  x_treino_poly = pf.transform(x_treino)
  x_teste_poly = pf.transform(x_teste)

  regressor_linear = LinearRegression()

  regressor_linear = regressor_linear.fit(x_treino_poly,y_treino)

  y_resposta_treino = regressor_linear.predict(x_treino_poly)
  y_resposta_teste = regressor_linear.predict(x_teste_poly)


  mse_in = mean_squared_error(y_treino,y_resposta_treino)
  rmse_in = math.sqrt(mse_in)
  r2_in = r2_score(y_treino,y_resposta_treino)
  rmspe = (np.sqrt(np.mean(np.square((y_teste - y_resposta_teste) / y_teste))))
  mse_out = mean_squared_error(y_teste,y_resposta_teste)
  rmse_out = math.sqrt(mse_out)
  r2_out = r2_score(y_teste,y_resposta_teste)

  print(' %3d %17.4f %15.4f  %15.4f' % ( k, rmse_in, rmse_out, rmspe))
```

Geração de CSV

In [ ]:

```
dados_embaralhados = dados.sample(frac=1,random_state=12345)
x = dados_embaralhados.drop(['preco'],axis = 1).to_numpy()
y = dados_embaralhados['preco'].to_numpy()

#x = dados.drop(['preco'])to_numpy()
#y = dados.to_numpy()
x_final = dados_teste.to_numpy()
#y_teste = dados_teste.to_numpy()

x_treino_final = x
y_treino_final = y

#teste
x_teste_final = x_final
```

Scaler para geração do CSV

In [ ]:

```
#escala = StandardScaler()
#escala.fit(x_treino)
```

```
#x_treino_final = escala.transform(x_treino)
#x_teste_final = escala.transform(x_teste)
```

KNN Regressor

In [ ]:

```
for k in range(1,5): #colocado um alcance curto do for para não tornar longa a execução f
inal do arquivo com tudo junto
  regressor_knn =KNeighborsRegressor(n_neighbors = k,weights ='distance')
  regressor_knn = regressor_knn.fit(x_treino,y_treino)

  y_resposta_treino = regressor_knn.predict(x_treino)
  y_resposta_teste = regressor_knn.predict(x_teste)

  mse_in = mean_squared_error(y_treino,y_resposta_treino)
  rmse_in = math.sqrt(mse_in)
  r2_in = r2_score(y_treino,y_resposta_treino)

  mse_out = mean_squared_error(y_teste,y_resposta_teste)
  rmse_out = math.sqrt(mse_out)
  r2_out = r2_score(y_teste,y_resposta_teste)

  rmspe = (np.sqrt(np.mean(np.square((y_teste - y_resposta_teste) / y_teste))))

  print(' %3d %17.4f %15.4f %15.4f  %15.4f' % (k, rmse_in, rmse_out,rmspe, r2_out ) )
```

Gerar CSV com KNN

In [ ]:

```
regressor_knn =KNeighborsRegressor(n_neighbors = 19,weights = 'distance')
regressor_knn = regressor_knn.fit(x_treino_final,y_treino_final)

y_resposta_treino = regressor_knn.predict(x_treino_final)
y_resposta_teste = regressor_knn.predict(x_teste_final)
```

In [ ]:

```
aux = pd.read_csv('conjunto_de_teste.csv')
resposta_regressor_KNN = pd.DataFrame({'Id':aux.pop('Id'), 'preco':np.squeeze((y_respost
a_teste))})
resposta_regressor_KNN.to_csv("resposta_regressorKNN.csv",index=False)
```

In [ ]:

```

```

Random Forest Regressor

Testando parâmetros

In [ ]:

```
regressor = RandomForestRegressor(bootstrap = True, max_depth = 15, max_features = 5, mi
n_samples_leaf = 2, min_samples_split = 5, n_estimators = 200, random_state = 0)
regressor.fit(x_treino,y_treino)
y_resposta_teste = regressor.predict(x_teste)
kfold  = KFold(n_splits=15, shuffle=True)
resultado = cross_val_score(regressor, x_treino, y_treino, cv = kfold)
print("K-Fold (R^2) Scores: {0}".format(resultado))
print("Media do R^2 por validação cruzada K-Fold: {0}".format(resultado.mean()))
```

In [ ]:

```
for n in range(1,16):
  regressor = RandomForestRegressor(n_estimators = n,random_state = 0)
  regressor.fit(x_treino,y_treino)
#  y_resposta_teste = regressor.predict(x_teste)

  kfold  = KFold(n_splits=15, shuffle=True)
  resultado = cross_val_score(regressor, x_treino, y_treino, cv = kfold)
  print("n = %i"%n)
  print("K-Fold (R^2) Scores: {0}".format(resultado))
  print("Media do R^2 por validação cruzada K-Fold: {0}".format(resultado.mean()))
```

Gerando CSV com o Random Forest

In [ ]:

```
regressor = RandomForestRegressor(bootstrap = True, max_depth = 15, max_features = 5, mi
n_samples_leaf = 2, min_samples_split = 5, n_estimators = 200, random_state = 0)
regressor.fit(x_treino_final,y_treino_final)
y_resposta_teste = regressor.predict(x_teste_final)


#r2_score(y_teste,y_RandomForest)
#y_resposta_teste
```

In [ ]:

```
aux = pd.read_csv('conjunto_de_teste.csv')
resposta_regressor_randomForest = pd.DataFrame({'Id':aux.pop('Id'), 'preco':np.squeeze((
y_resposta_teste))})
resposta_regressor_randomForest.to_csv("resposta_randomforest.csv",index=False)
```