Curso de Data Science

Prof. MSc. Eng. Marcelo Bianchi

Trabalho 1 – Tipo 2 - Curso de Data Science

Integrantes do Grupo:

Nome: Carlos Rios Jr.

Nome: Lenilson Florencio

Nome: Nara Guimarães

Nome: Vanessa Kaplum Foleis

1) (5,0 pontos) Regressão Linear Simples

Dado o dataset abaixo, calcule a regressão linear simples programando em Python

Horas_de_Academia Semana	Nota Teste Aptidão Fisica
1	53
5	74
7	59

8	43
10	56
11	84
14	96
15	69
15	84
19	83

Predição: Nota Teste Aptidão Física

1) **(0,5 ponto) Importar o Data Set e aplicar a técnica Missing Data**

   Resposta: Os dados presentes nesse arquivo foram salvos em um arquivos do tipo .csv denominado "dataset1-teste-aptidao.csv". Em seguida, transformamos o arquivo .csv em um dataset denominado "df1", por meio da função "read_csv" da biblioteca Pandas. Conforme comando abaixo:

```
df1 = pd.read_csv('dataset1-teste-aptidao.csv', sep=';')
```

   Os dados avaliados não possuem dados nulos ou faltantes, por isso não é necessário realizar a técnica de "Missing Data". Comprovamos que não há dados faltantes por meio da função "is_null" da biblioteca Pandas. Conforme comando abaixo:

```
df1.isnull().sum()
```

2) **(0,5 ponto) Dividir o dataset entre o Training Set e o Test Set.**

   Resposta: O dataset foi divido em Training Set e em Testing Set por meio da função "train_test_split" presente no módulo model_selection da biblioteca sckikit-learn. De forma que 80% dos dados foram atribuídos ao grupo de treino (X_train e y_train) e 20% dos dados foram atribuídos ao grupo de teste (X_test e y_test). A alocação dos dados no grupo treino e no grupo teste objetivou uma correta generalização do modelo de regressão linear simples. Além de este ser um valor recomendado pelo princípio de Pareto (fonte: https://www.kdnuggets.com/2019/03/pareto-principle-data-scientists.html). Conforme comandos abaixo:

```
X = df1.iloc[:, :-1].values
y = df1.iloc[:, 1].values


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

3) (0,5 ponto) Aplicar Feature Scaling (Se for aplicável, se não for então justificar)

Resposta: No presente data set, não há necessidade de realizar a técnica de "Feature Scaling", visto que os dados não estão em ordem de grandeza diferenciada, ou seja os dados não estão dispersos de forma relevante.

4) (0,5 ponto) Aplicar Dummy Variable (Se for aplicável, se não for então justificar)

Resposta: Não é necessário, diante do tipo de dados (Não há dados categóricos)

5) (0,5 ponto) Aplicar a Simple Linear Regression

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

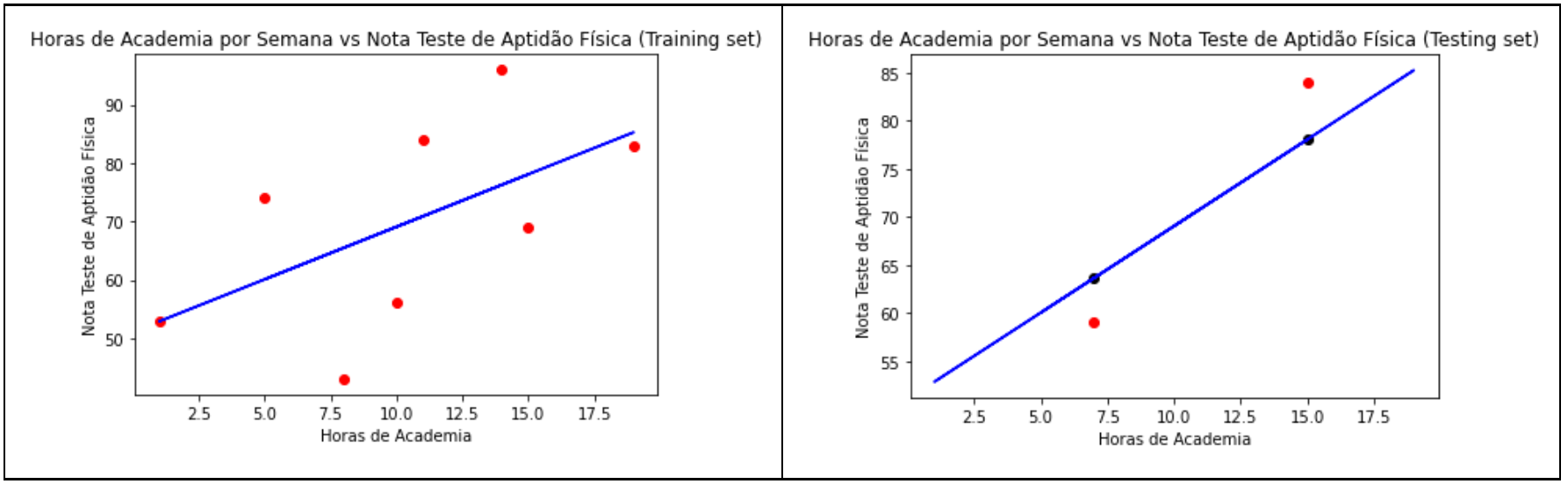
6) (0,5 ponto) Construir o Gráfico (Scatter Plot)

```
# Gráfico do Training Set
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
```

```
plt.title('Horas de Academia por Semana vs Nota Teste de Aptidão Física (Training set)')
plt.xlabel('Horas de Academia')
plt.ylabel('Nota Teste de Aptidão Física')
plt.show()
```

Gráfico do Testing Set

```
y_pred = regressor.predict(X_test)

plt.scatter(X_test, y_test, color = 'red')
plt.scatter(X_test, y_pred, color='black')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Horas de Academia por Semana vs Nota Teste de Aptidão Física (Testing set)')
plt.xlabel('Horas de Academia')
plt.ylabel('Nota Teste de Aptidão Física')
plt.show()
```



7) (0,5 ponto) Criar a tabela no banco de dados SQLite

```
con = sq3.connect("avaliacao1.db")
df1.to_sql("Teste", con, index = False)
con.execute("SELECT name FROM sqlite_master WHERE type='table' ORDER BY name").fetchall()
```

8) (0,5 ponto) Aplicar uma consulta em linguagem SQL que irá trazer uma listagem da tabela

```
pd.read_sql("Select * FROM Teste",con)
```

9) (0,5 ponto) Apresentação e explicação do exercício ao professor

10) (0,5 ponto) Responder uma dúvida ou questão do professor

2) (5,0 pontos) Regressão Linear Múltipla

Dado o dataset abaixo, calcule a regressão linear múltipla programando em python

CarName	fueltype	aspiration	carbody	enginesize	horsepower	peakrpm	Car price
alfa-romero giulia	gas	std	convertible	130	111	5000	13495
alfa-romero stelvio	gas	std	convertible	130	111	5000	16500
alfa-romero Quadrifoglio	gas	std	hatchback	152	154	5000	16500
audi 100 ls	gas	std	sedan	109	102	5500	13950
audi 100ls	gas	std	sedan	136	115	5500	17450
audi fox	gas	std	sedan	136	110	5500	15250
audi 100ls	gas	std	sedan	136	110	5500	17710
audi 5000	gas	std	wagon	136	110	5500	18920
audi 4000	gas	turbo	sedan	131	140	5500	23875
audi 5000s (diesel)	gas	turbo	hatchback	131	160	5500	17859167
bmw 320i	gas	std	sedan	108	101	5800	16430
bmw 320i	gas	std	sedan	108	101	5800	16925
bmw x1	gas	std	sedan	164	121	4250	20970
bmw x3	gas	std	sedan	164	121	4250	21105
bmw z4	gas	std	sedan	164	121	4250	24565
bmw x4	gas	std	sedan	209	182	5400	30760
bmw x5	gas	std	sedan	209	182	5400	41315

bmw x3	gas	std	sedan	209	182	5400	36880
chevrolet impala	gas	std	hatchback	61	48	5100	5151
chevrolet monte carlo	gas	std	hatchback	90	70	5400	6295
chevrolet vega 2300	gas	std	sedan	90	70	5400	6575
dodge rampage	gas	std	hatchback	90	68	5500	5572

Predição: Car Price

1) **(0,5 ponto) Importar o Dataset e aplicar a técnica Missing Data**

Resposta: Para importação do dataset, salvamos a tabela no formato csv e fizemos a importação por meio do seguinte comando:

```
df2 = pd.read_csv('dataset2.csv', sep=';')
```

O sep, de separador foi utilizado porque nosso excel estava configurado com separador ";". Caso o separador já fosse uma vírgula "," não haveria a necessidade de incluir esse argumento.

Como não havia missing data no nosso dataset, não foi preciso aplicar essa técnica.

2) **(0,5 ponto) Dividir o dataset entre o Training Set e o Test Set**

```
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)
```

Optamos por fazer 20% teste e 80% treino pois assim conseguimos fazer o treinamento do modelo sem haver um overfitting. Além de este ser um valor recomendado pelo princípio de Pareto (fonte: https://www.kdnuggets.com/2019/03/pareto-principle-data-scientists.html)

3) (0,5 ponto) Aplicar Feature Scaling (Se for aplicável, se não for então justificar)

Optamos por aplicar o feature scaling uma vez que os valores se apresentavam com ordem de grandeza diferenciada. Para evitar que o modelo considerasse que uma certa variável com valores maiores possuísse uma importância maior que as demais, aplicou-se essa técnica a fim de garantir que todas estivessem na mesma escala de grandeza.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_car_arrumado_fitScale = pd.DataFrame(scaler.fit_transform(df_car_arrumado))
df_car_arrumado_fitScale
```

4) (0,5 ponto) Aplicar Dummy Variable (Se for aplicável, se não for então justificar)

Resposta: foram utilizadas Dummy Variables para a feature *carbody*. Quanto a feature *aspiration*, substitui-se por números (0 e 1), considerando que as entradas eram binárias (*std* ou *turbo*). Além disso, extraiu-se a marca do nome do carro (*CarName*) e foram criadas Dummy Variables para esta nova feature também. A coluna fueltype foi excluída, uma vez que esta possuia o mesmo valor para todas as linhas, não sendo portanto relevante para o modelo.

Para as duas variáveis que aplicamos a dummy variables, foram também aplicadas a técnica de dummy trap, removendo uma coluna de cada conjunto das dummy variables criadas, ou seja, para o carbody e carname(após extração da marca)

5) (0,5 ponto) Aplicar a Multiple Linear Regression com a técnica de Backward Elimination

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

6) (0,5 ponto) Construir o Gráfico (Scatter Plot)

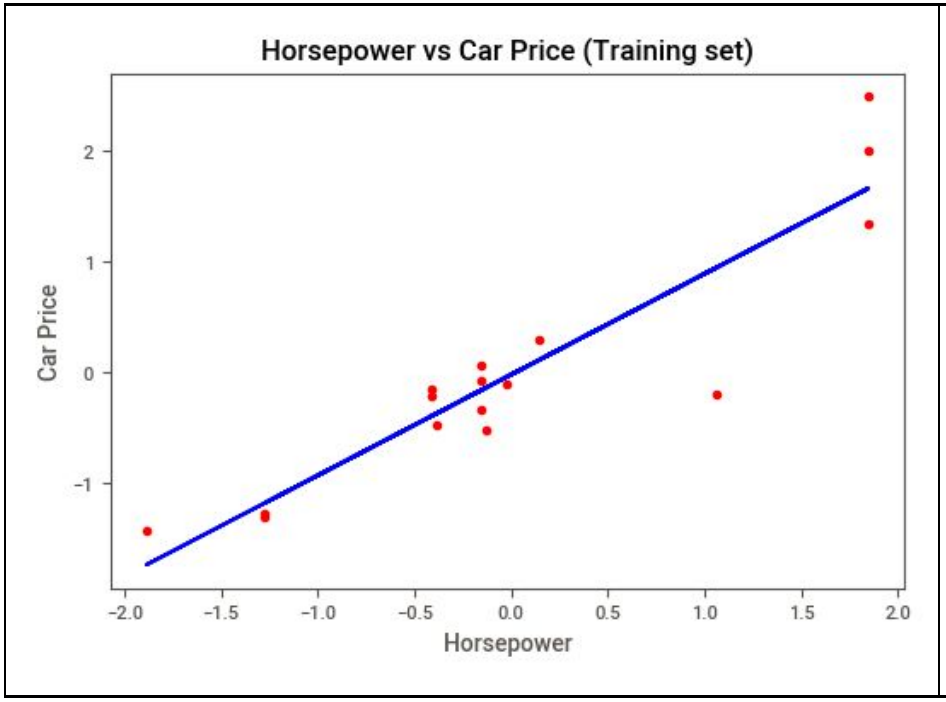
Resposta: Utilizando a técnica do Backward Elimination (fonte: https://medium.com/@mayankshah1607/machine-learning-feature-selection-with-backward-elimination-955894654026) chegamos as duas features que melhor afetam a predição, que são *horsepower* e *enginesize*

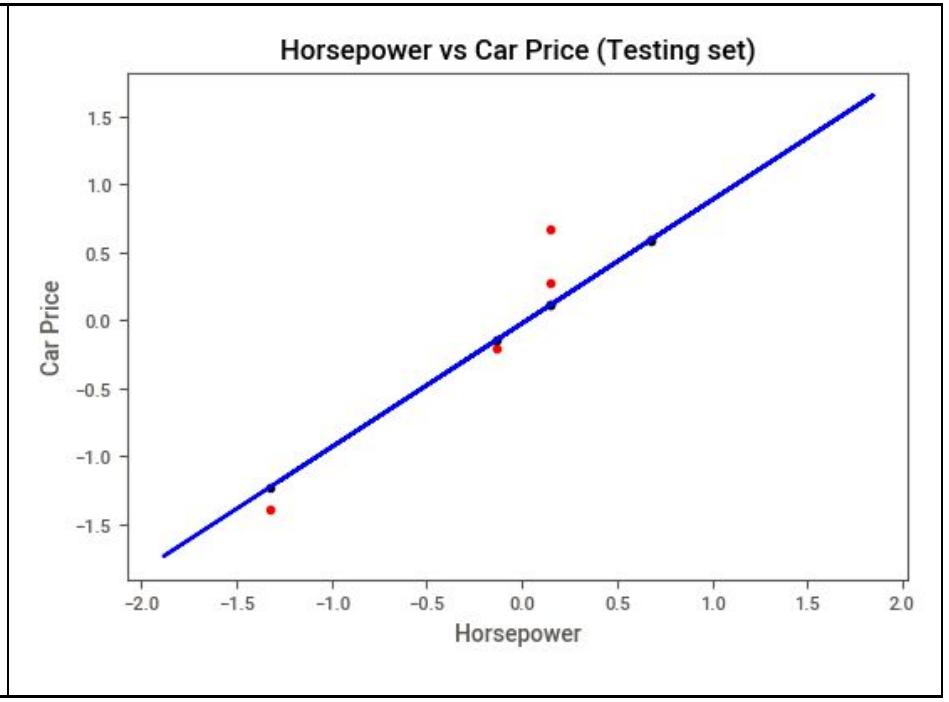
Nós também utilizamos a biblioteca sweetwiz (font: https://pypi.org/project/sweetviz/) e pela mesma, ambas as variáveis eram consideradas relevantes.

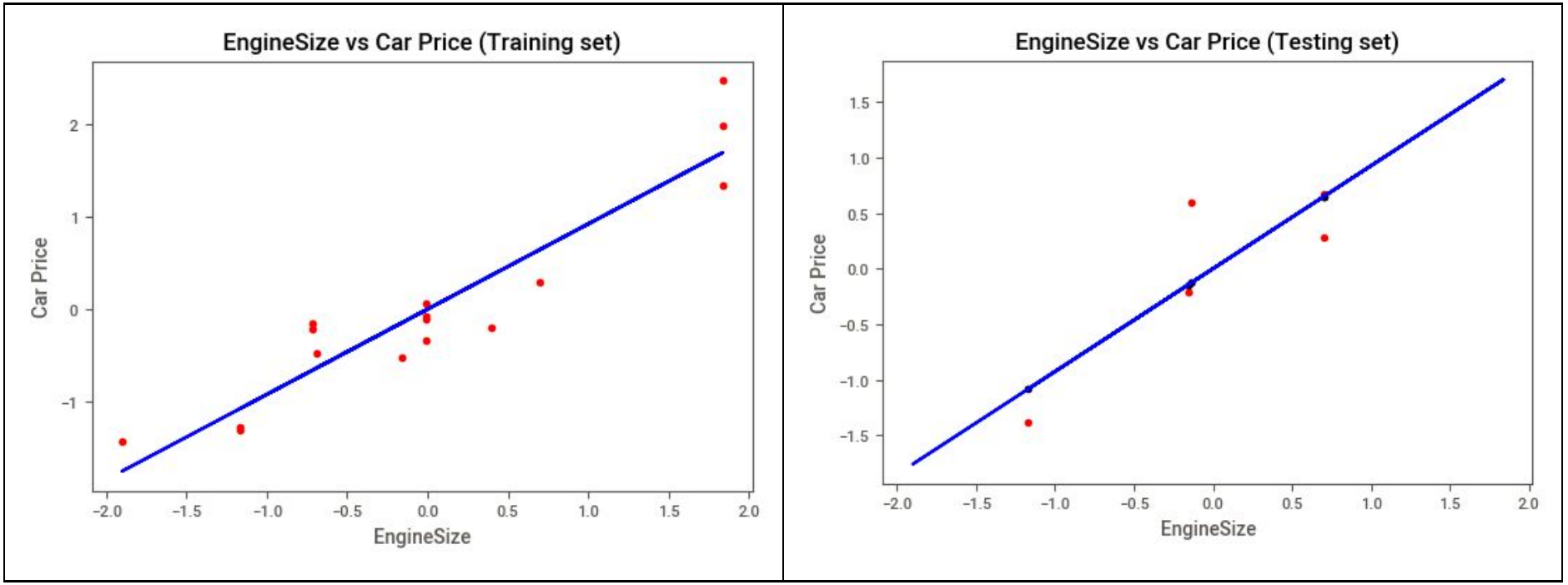
Abaixo os gráficos relativos a estas duas features, para o dataset de treino e de teste:

```
# Horsepower - Training Set
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Horsepower vs Car Price (Training set)')
plt.xlabel('Horsepower')
plt.ylabel('Car Price')
plt.show()

# Horsepower - Testing Set
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.scatter(X_test,y_pred, color = 'black') #plotar a predição
plt.title('Horsepower vs Car Price (Testing set)')
plt.xlabel('Horsepower')
plt.ylabel('Car Price')
plt.show()

# Enginepower - Training Set
plt.scatter(X_train, y_train, color = 'red')
```

```
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('EngineSize vs Car Price (Training set)')
plt.xlabel('EngineSize')
plt.ylabel('Car Price')
plt.show()

# Enginepower - Testing Set
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.scatter(X_test,y_pred, color = 'black') #plotar a predição
plt.title('EngineSize vs Car Price (Testing set)')
plt.xlabel('EngineSize')
plt.ylabel('Car Price')
plt.show()
```


Horsepower vs Car Price (Training set)
Car Price
Horsepower
2
1
0
−1
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Horsepower vs Car Price (Testing set)
Car Price
Horsepower
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



7) (0,5 ponto) Criar a tabela no banco de dados SQLite

```
import sqlite3 as sq3
con = sq3.connect("avaliacao1.db")
car_df.to_sql("CarPrice", con, index=False)
```

8) **(0,5 ponto) Aplicar uma consulta em linguagem SQL que irá trazer uma listagem da tabela**

```
car_price = pd.read_sql("SELECT * FROM CarPrice", con)
```

9) **(0,5 ponto) Apresentação e explicação do exercício ao professor**
10) **(0,5 ponto) Responder uma dúvida ou questão do professor**