IE 345 - K “Introduction to Deep Learning: Fundamentals Concepts”

Prof. Yuzo

Simple Linear Regression

pg. 60 - 63

In [1]:

```
# Importing the libraries
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:

```
# Importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
dataset.head()
```

Out[2]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [3]:

```
# Splitting the dataset into the Training set and 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 = 1/3, rando
m_state = 0)
```

In [4]:

```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[4]:

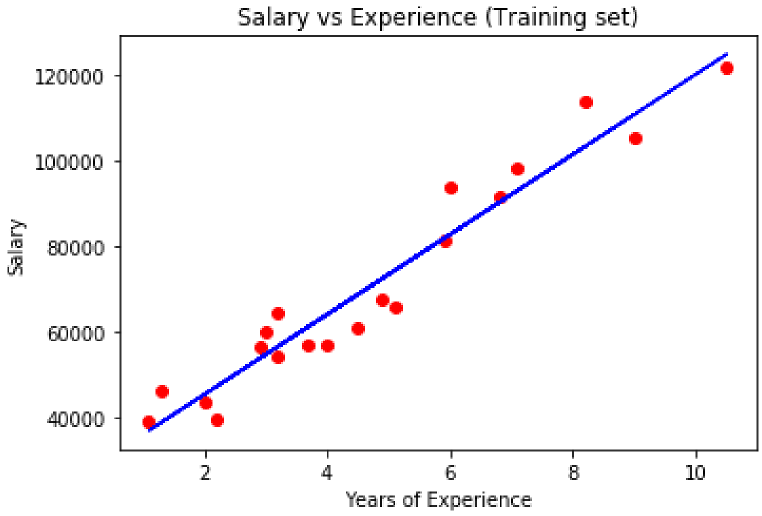
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [5]:

```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
```

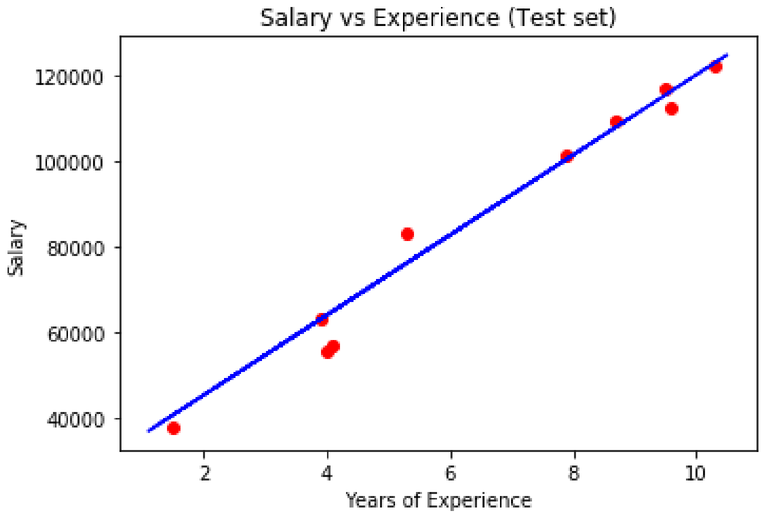
In [6]:

```python
# Visualising the Training set results
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



In [7]:

```python
# Visualising the Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



Multiple Linear Regression

pg. 65 - 71

In [8]:

```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [9]:

```
# Importing the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
#Let’s look at the data:
dataset.head()
```

Out[9]:

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [10]:

```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X = onehotencoder.fit_transform(X).toarray()

# Avoiding the Dummy Variable Trap
X = X[:, 1:]
print(X[0:12, 1:].astype(int))
```

```
[[     1 165349 136897 471784]
 [     0 162597 151377 443898]
 [     0 153441 101145 407934]
 [     1 144372 118671 383199]
 [     0 142107  91391 366168]
 [     1 131876  99814 362861]
 [     0 134615 147198 127716]
 [     0 130298 145530 323876]
 [     1 120542 148718 311613]
 [     0 123334 108679 304981]
 [     0 101913 110594 229160]
 [     0 100671  91790 249744]]

C:\Users\pablo\Python\envs\DAVID\lib\site-packages\sklearn\preprocessing\_encod
ers.py:371: FutureWarning: The handling of integer data will change in version
0.22. Currently, the categories are determined based on the range [0, max(value
s)], while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "cat
egories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the catego
ries to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\pablo\Python\envs\DAVID\lib\site-packages\sklearn\preprocessing\_encod
ers.py:392: DeprecationWarning: The 'categorical_features' keyword is deprecate
d in version 0.20 and will be removed in 0.22. You can use the ColumnTransforme
r instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

In [11]:

```
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, rando
m_state = 0)
```

In [12]:

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

Out[12]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [13]:

```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
print('y_pred: ',y_pred)
```

```
y_pred:  [103015.20159796 132582.27760816 132447.73845175  71976.09851259
 178537.48221054 116161.24230163  67851.69209676  98791.73374688
 113969.43533012 167921.0656955 ]
```

Pablo David Minango Negrete

pablodavid218@gmail. com

Lisber Arana Hinostroza

lisberarana@gmail. com