Linear Regresssion on large data

Case: Predicting the profit from R&D Spend,Administration,Marketing Spend,State

1. Importing libraries
2. load data
3. clean data(nulls,duplicate)
4. preprocess(encoding,scaling)
5. Split data
6. Create and train model
7. Test the model
8. Evaluation

1. Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import
r2_score,mean_absolute_error,mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

1. load data

```
df=pd.read_csv(r"C:\Mypythonfiles\50_Startups.csv")
df.head()
```

	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

clean data

```
df.isnull().sum()
```

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

```
df.drop_duplicates(inplace =True)

df.head()
```

```
   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


   State_encoded
0              2
1              0
2              1
3              2
4              1
```

preprocess

```
s_e = LabelEncoder()
df['State_encoded'] = s_e.fit_transform(df['State'])
df.head()
```

```
   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


   State_encoded
0              2
1              0
2              1
3              2
4              1
```

Split - ind,dep

```
x = df[['R&D Spend','Administration','Marketing
Spend','State_encoded']]
y = df['Profit']
```

Split - train and test

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size =
0.2, random_state=42)
```

Create and train

```
profit_model = LinearRegression()
profit_model.fit(x_train,y_train)

LinearRegression()
```

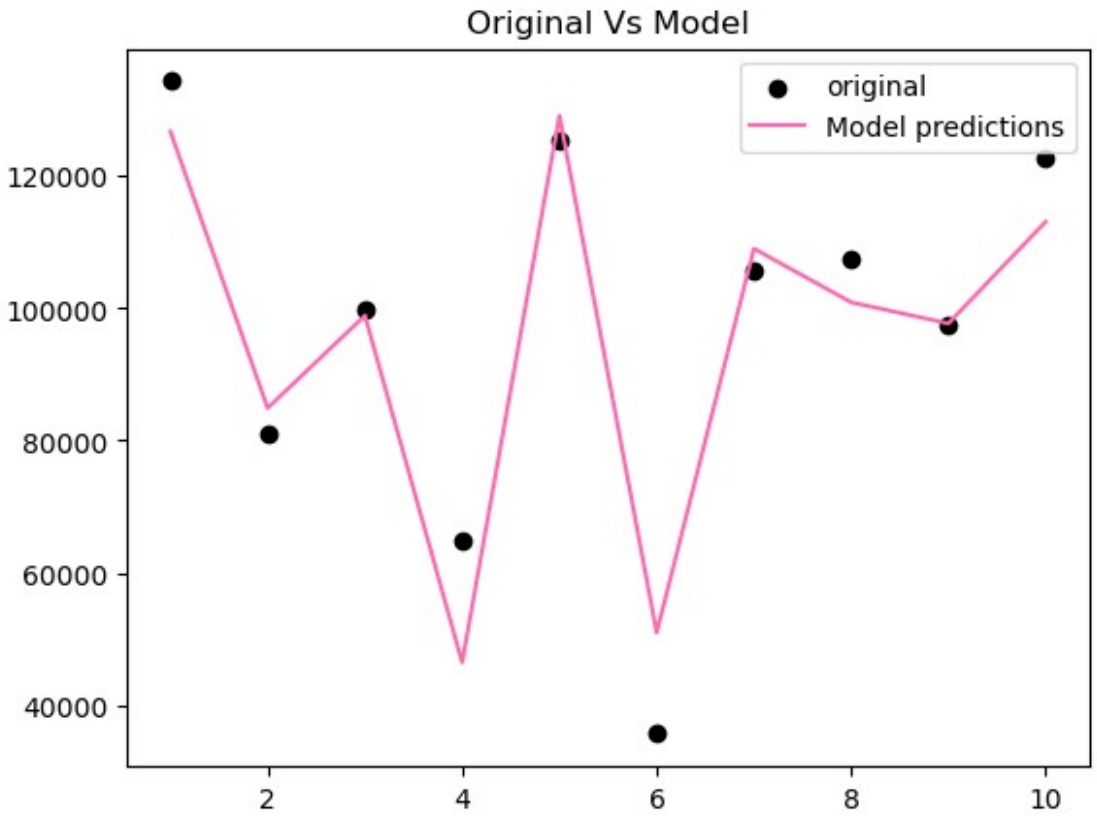
Test

```
Rd = float(input("Enter your R&D Expence: "))
a = float(input("Enter your Administrative Expence: "))
Ms = float(input("Enter your Marketing Expence : "))
St = input("Enter your State : ")

Enter your R&D Expence:  16000
Enter your Administrative Expence:  91391
Enter your Marketing Expence :  36676
Enter your State :  New York

sta_enc = s_e.transform([St])[0]
print(sta_enc)

2

result = profit_model.predict([[Rd,a,Ms,sta_enc]]) #passing
independent variable(time in 2D)
print("the predicted state is : ",result[0])

the predicted state is :  61847.92035126767

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\base.py:439:
UserWarning: X does not have valid feature names, but LinearRegression
was fitted with feature names
  warnings.warn(
```

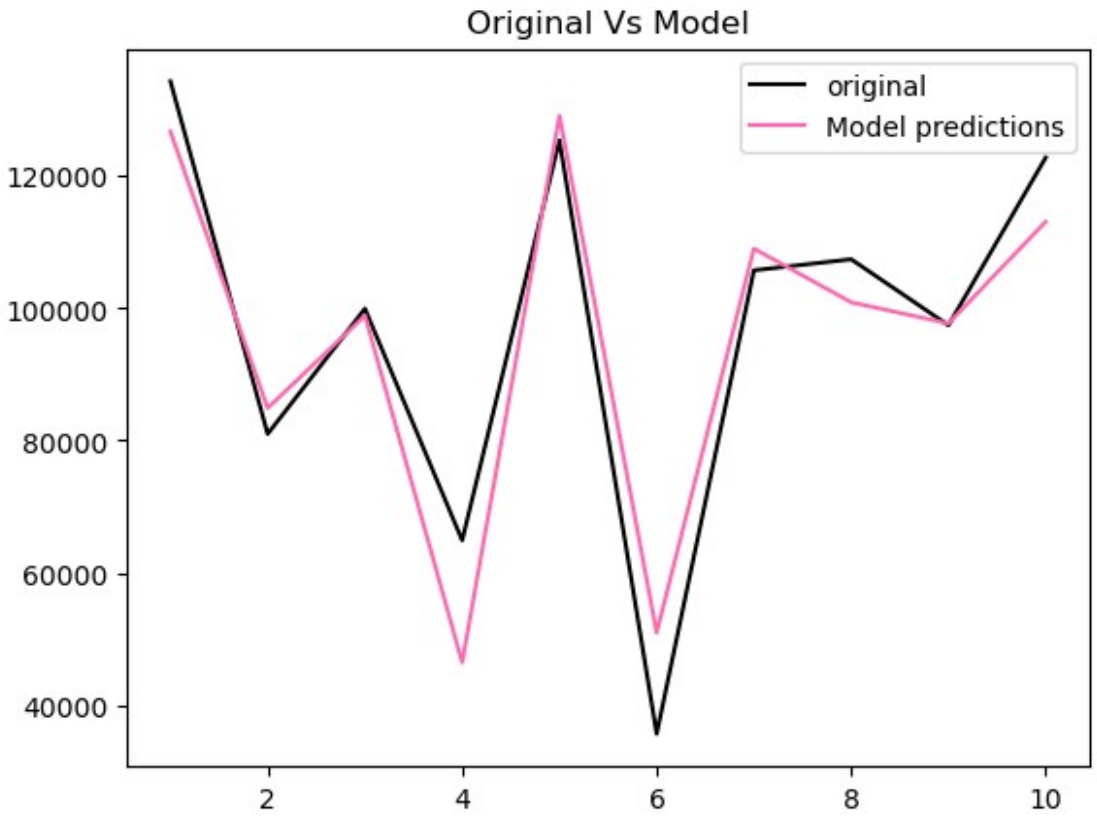
Evaluation:

- Predict test values
- visualize
- metrics

```
model_predictions = profit_model.predict(x_test)

len(y_test)

10

len(x_test)

10

plt.scatter(np.arange(1,11), y_test, color = 'k', label= 'original')
plt.plot(np.arange(1,11),model_predictions, color = 'hotpink', label =
'Model predictions')
plt.title("Original Vs Model")
plt.legend()
plt.show()
```




```
plt.plot(np.arange(1,11), y_test, color = 'k', label= 'original')
plt.plot(np.arange(1,11),model_predictions, color = 'hotpink', label =
'Model predictions')
plt.title("Original Vs Model")
plt.legend()
plt.show()
```




```
r2score = r2_score(y_test, model_predictions)
print(r2score)
if r2score > 0.5:
    print("Model is good")
else:
     print("Model is not good")
```

```
0.9000614254946402
Model is good
```

```
mse = mean_squared_error(y_test,model_predictions)
print(mse)
```

```
80929465.49097784
```

```
mae = mean_absolute_error(y_test,model_predictions)
print(mae)
```

```
6979.17574672139
```