In [279]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score,mean_squared_error,r2_score
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import seaborn as sns
import statsmodels.formula.api as smf
from sklearn.linear_model import LogisticRegression
```

In [280]:
```
#for 3-Dimensional representation
from mpl_toolkits.mplot3d.axes3d import get_test_data
# This import registers the 3D projection, but is otherwise unused.
from mpl_toolkits.mplot3d import Axes3D
```

In [281]:
```
#reading data from CSV
data = pd.read_csv("advertising.csv")
```

In [282]:
```
#to display the top 5 instances in the dataframe
data.head()
```

Out[282]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [283]:
```
#to check the columns present in the dataframe
data.columns
```

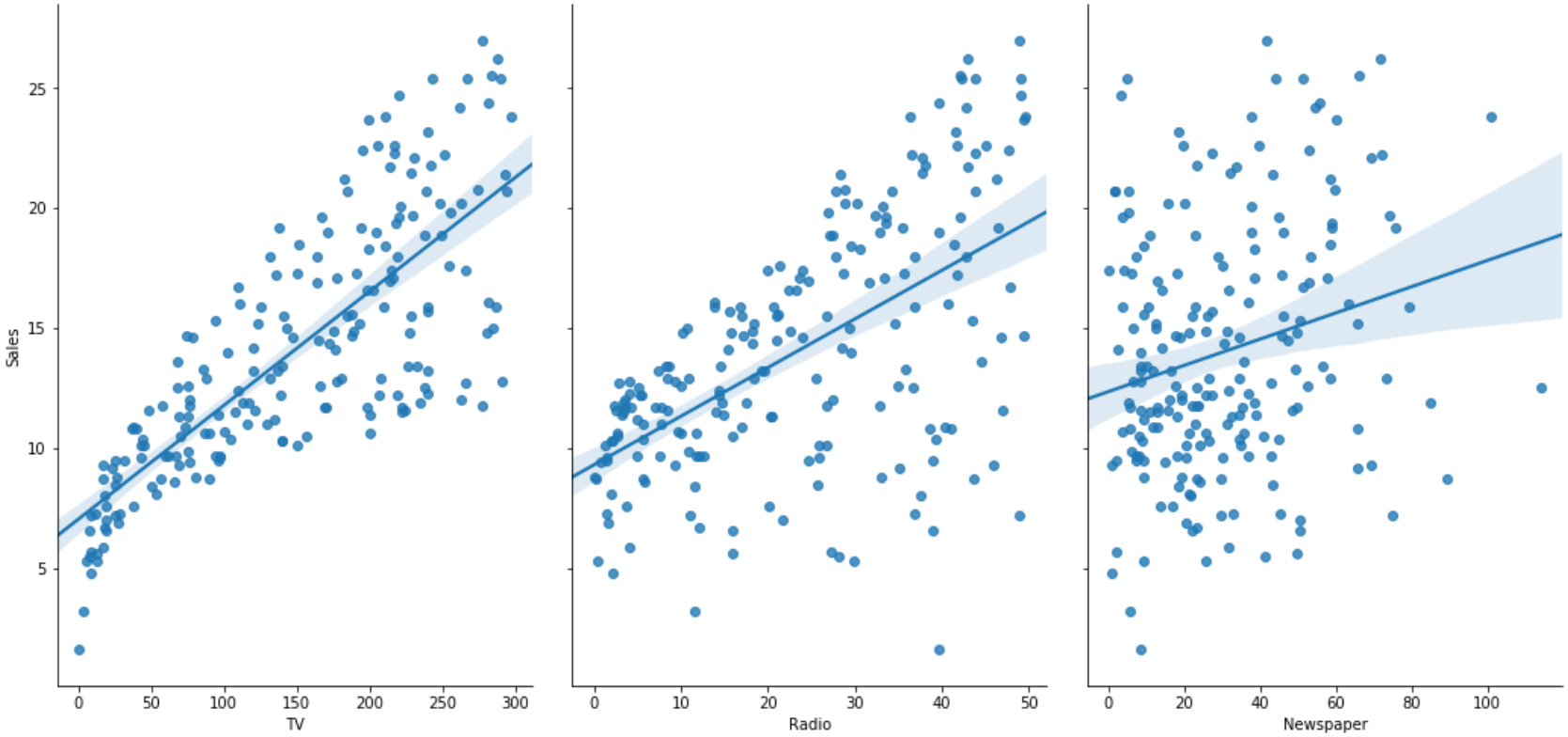
Out[283]:
```
Index(['Unnamed: 0', 'TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [284]:
```
#dropping the unwanted columns in the dataframe
data=data.drop(['Unnamed: 0'], axis=1)
```

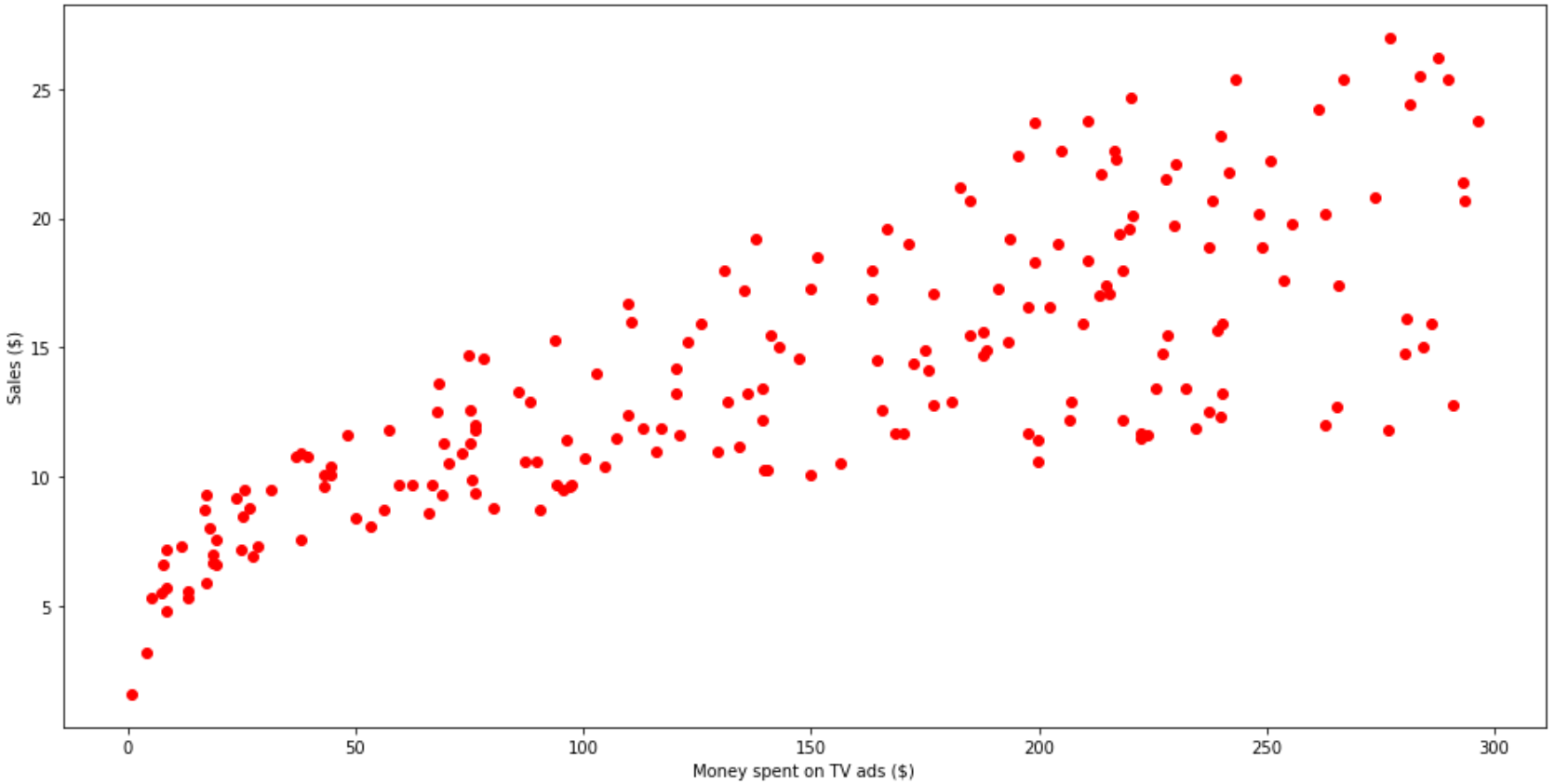
In [285]:
```
sns.pairplot(data, x_vars=['TV','Radio','Newspaper'], y_vars='Sales', size=7, aspect=0.7, kind='reg')
```

```
/Applications/anaconda2/envs/py36/lib/python3.5/site-packages/seaborn/axisgrid.py:2065: UserWarning:
The `size` parameter has been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
/Applications/anaconda2/envs/py36/lib/python3.5/site-packages/scipy/stats/stats.py:1713: FutureWarni
ng: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` in
stead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`,
which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[285]:
```
<seaborn.axisgrid.PairGrid at 0x1c18510630>
```



In [286]:
```
#mentioning the size of the plot/figure.
plt.figure(figsize=(16, 8))
#mentioning the scatter function to consider which columns in the dataframe
plt.scatter(
    data['TV'],
    data['Sales'],
    c='red'
)
#Labeling both X & Y axis's
plt.xlabel("Money spent on TV ads ($)")
plt.ylabel("Sales ($)")
#To display the scatter plot with above mentioned attributes
plt.show()
```



In [287]:
```
#reshape allows to change the shape of the dataframe without loosing/changing data
X = data['TV'].values.reshape(-1,1)
y = data['Sales'].values.reshape(-1,1)

#getting the Linear Regression function from (from sklearn.linear_model import LinearRegression)
reg = LinearRegression()
#
reg.fit(X, y)
```

Out[287]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [288]:
```
#accuracy of sales prediction when only TV add's investments in considered
#Returns the coefficient of determination R^2 of the prediction.
reg.score(X,y)
```

Out[288]:
```
0.611875050850071
```

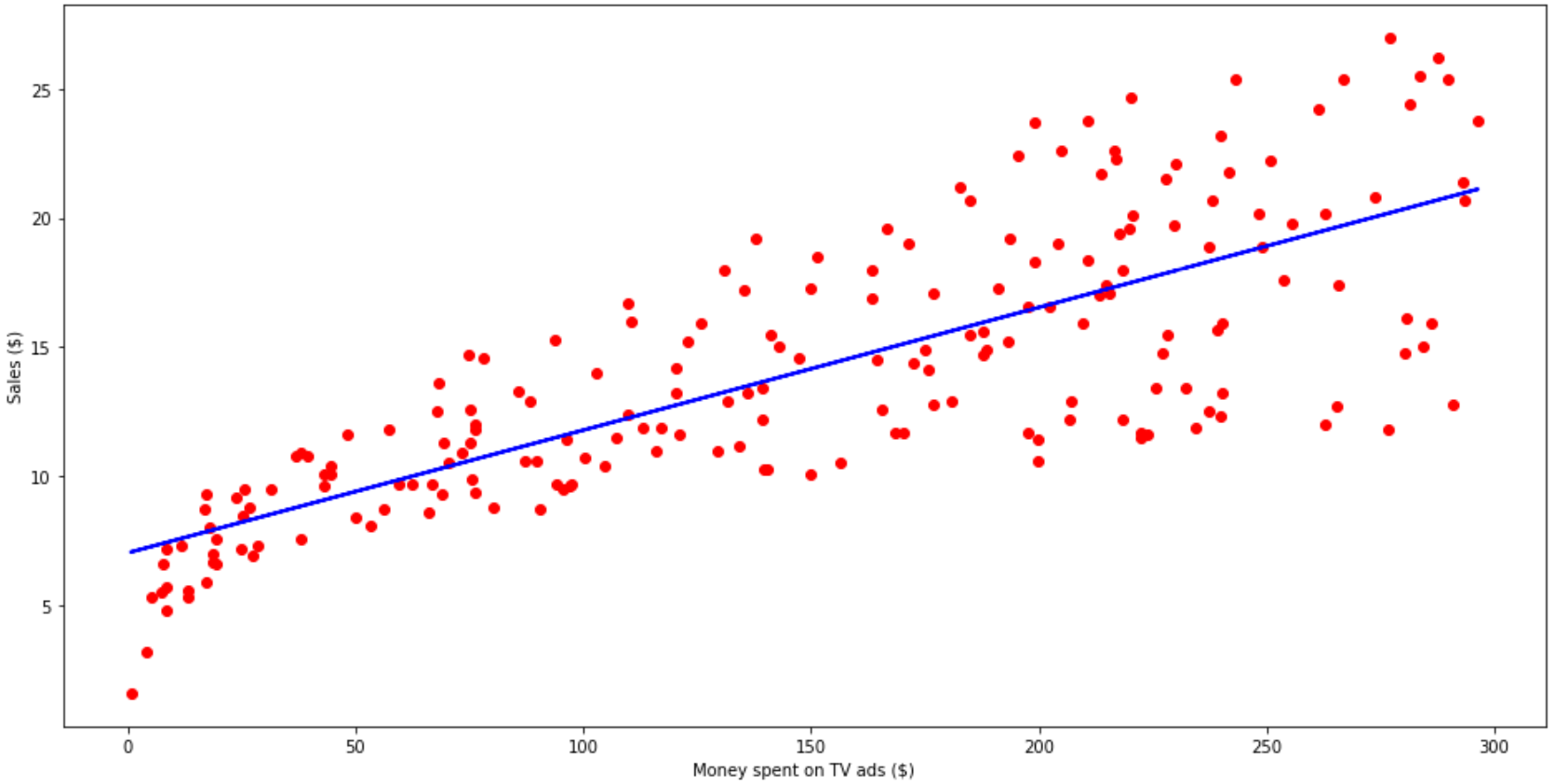
In [289]:
```
print(reg.coef_[0][0])
print(reg.intercept_[0])

print("The linear model is: Y = {:.5} + {:.5}X".format(reg.intercept_[0], reg.coef_[0][0]))
```

```
0.047536640433019764
7.032593549127693
The linear model is: Y = 7.0326 + 0.047537X
```

In [290]:
```
predictions = reg.predict(X)
plt.figure(figsize=(16, 8))
plt.scatter(
    data['TV'],
    data['Sales'],
    c='red'
)
plt.plot(
    data['TV'],
    predictions,
    c='blue',
    linewidth=2
)

plt.xlabel("Money spent on TV ads ($)")
plt.ylabel("Sales ($)")
plt.show()
```


Sales ($)
Money spent on TV ads ($)
0
50
100
150
200
250
300
5
10
15
20
25

In [291]:
```
#differenct statistical values regarding the model on the dataset.

X = data['TV']
y = data['Sales']

X2 = sm.add_constant(X)
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary()) #prints the entire summary of the dataset.
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.612
Model:                            OLS   Adj. R-squared:                  0.610
Method:                 Least Squares   F-statistic:                     312.1
Date:                Thu, 17 Jan 2019   Prob (F-statistic):           1.47e-42
Time:                        02:00:48   Log-Likelihood:                -519.05
No. Observations:                 200   AIC:                             1042.
Df Residuals:                     198   BIC:                             1049.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.0326      0.458     15.360      0.000       6.130       7.935
TV             0.0475      0.003     17.668      0.000       0.042       0.053
==============================================================================
Omnibus:                        0.531   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.767   Jarque-Bera (JB):                0.669
Skew:                          -0.089   Prob(JB):                        0.716
Kurtosis:                       2.779   Cond. No.                         338.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [292]:
```
#NOW CONSIDERING INVESTMENTS ON BOTH TV AND NEWSPAPER  TO PREDICT SALES, SO REMOVING SALES, UNNAMED A
ND RADIO
#FROM THE DATASET, WHICH KEEPS ONLY TV AND NEWSPAPER AS ATTRIBUTES IN 'X'

Xs = data.drop(['Sales','Radio'], axis=1)
Xs.head()
#y = data['Sales'].reshape(-1,1)

reg1 = LinearRegression()
reg1.fit(Xs, y)
```

Out[292]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [293]:
```
#intercept and co-efficient for multiple regression

print(reg1.coef_)
print(reg1.intercept_)
```

```
[0.04690121 0.04421942]
5.774947967911631
```

In [296]:
```
print("The linear model is: Y = {:.5} + {:.5}*TV +{:.5}*newspaper".format(reg1.intercept_, reg1.coef_
[0], reg1.coef_[1]))
```

```
The linear model is: Y = 5.7749 + 0.046901*TV +0.044219*newspaper
```

In [297]:
```
#Returns the coefficient of determination R^2 of the prediction.
reg1.score(Xs, y)
```

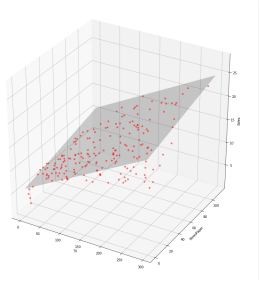
Out[297]:
```
0.6458354938293271
```

In [298]:
```
df2 = data.drop(['Radio'], axis=1)
```

In [299]:
```
model = smf.ols(formula='Sales ~ TV + Newspaper', data=df2)
results_formula = model.fit()
```

In [300]:
```
x_surf, y_surf = np.meshgrid(np.linspace(data.TV.min(), data.TV.max(), 300),np.linspace(data.Newspape
r.min(), data.Newspaper.max(), 100))
onlyX = pd.DataFrame({'TV': x_surf.ravel(), 'Newspaper': y_surf.ravel()})
fittedY=results_formula.predict(exog=onlyX)
```

In [301]:
```
fig = plt.figure(figsize=(16, 16))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(data['TV'],data['Newspaper'],data['Sales'],c='red', marker='o', alpha=0.5)
ax.plot_surface(x_surf,y_surf,fittedY.values.reshape(100,300), color='black', alpha=0.19)
ax.set_xlabel('TV')
ax.set_ylabel('NewsPaper')
ax.set_zlabel('Sales')
plt.show()
```



In [302]:
```
data_lgr = pd.read_csv("advertising.csv")
```

In [303]:
```
xdflgr=data_lgr.drop(['Unnamed: 0','Sales'], axis=1)
xdflgr.head()
```

Out[303]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

In [304]:
```
ydflgr=dflgr['Sales']
ydflgr.head()
```

Out[304]:
```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: Sales, dtype: float64
```

In [305]:
```
len(Xlgr)
```

Out[305]:
```
200
```

In [306]:
```
X_train,X_test,y_train,y_test=train_test_split(xdflgr,ydflgr,test_size=0.3,random_state=32)
```

In [307]:
```
#Logistic Regresion
log_clf = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial')
```

In [ ]:
```

```