In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("C:\\Users\\hp\\Python Data Analysis Course\\Python Course\\08-Linear-Regression-Models\\Advertising.csv"
```

In [3]:

```
df.head()
```

Out[3]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

Simple linear regression

In [4]:

```
X = df.drop('sales', axis=1)
```

In [6]:

```
y = df['sales']
```

In [7]:

```
from sklearn.model_selection import train_test_split
```

In [8]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

In [9]:

```
from sklearn.linear_model import LinearRegression
```

In [10]:

```
model = LinearRegression()
```

In [11]:

```
model.fit(X_train,y_train)
```

Out[11]:

```
▾ LinearRegression
LinearRegression()
```

In [12]:

```
test_prediction = model.predict(X_test)
```

In [13]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [14]:

```
MAE = mean_absolute_error(y_test,test_prediction)
```

In [20]:

```
RSME = np.sqrt (mean_squared_error(y_test,test_prediction))
```

In [16]:

```
MAE
```

Out[16]:

```
1.5116692224549084
```

In [21]:

```
RSME
```

Out[21]:

```
1.9485372043446383
```

Polynomial Model

In [22]:

```
from sklearn.preprocessing import PolynomialFeatures
```

In [ ]:

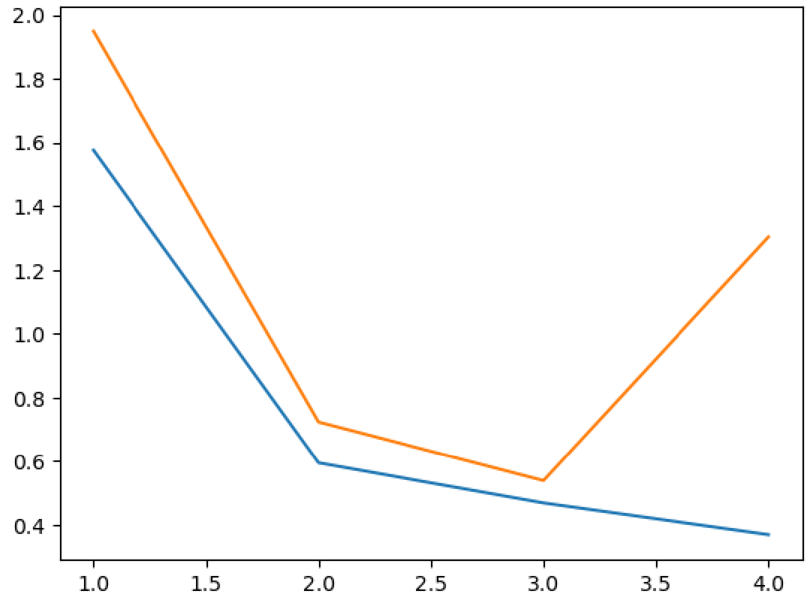
```
# This wewant to prevent overfitting and underfitting while finding the degree for which test RSME will be least.
# We can do this by creating a for loop and find RSME for a cretain range and
# compare to know which degree gives the best fit.
```

In [38]:

```
train_rsme_err = []
test_rsme_err = []

for d in range(1,5):
    polynomial_converter = PolynomialFeatures(degree=d, include_bias=False)
    poly_features = polynomial_converter.fit_transform(X)

    X_train, X_test, y_train, y_test = train_test_split(poly_features, y, test_size=0.30, random_state=42)

    model = LinearRegression()

    model.fit(X_train,y_train)

    train_pred = model.predict(X_train)
    test_pred = model.predict(X_test)

    train_err = np.sqrt(mean_squared_error(y_train,train_pred))
    test_err = np.sqrt(mean_squared_error(y_test, test_pred))

    train_rsme_err.append(train_err)
    test_rsme_err.append(test_err)
```

In [39]:

```
plt.plot(range(1,5), train_rsme_err)
plt.plot(range(1,5), test_rsme_err);
```



In [40]:

```
test_rsme_err
```

Out[40]:

```
[1.9485372043446383,
 0.7233218473857531,
 0.5392350985609965,
 1.3032265967218177]
```

In [ ]: