In this Assignment I will work on Advertising dataset. It includes TV, radio, newspaper(independents), and sales(dependent / target) columns. I will use Linear Regression model to predict sales.

I started with importing libraries and reading the data set.

```
[1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split, GridSearchCV
import warnings
warnings.filterwarnings('ignore')
```

```
[2]: df=pd.read_csv("Advertising.csv")
```

```
[3]: df.head()
```

[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

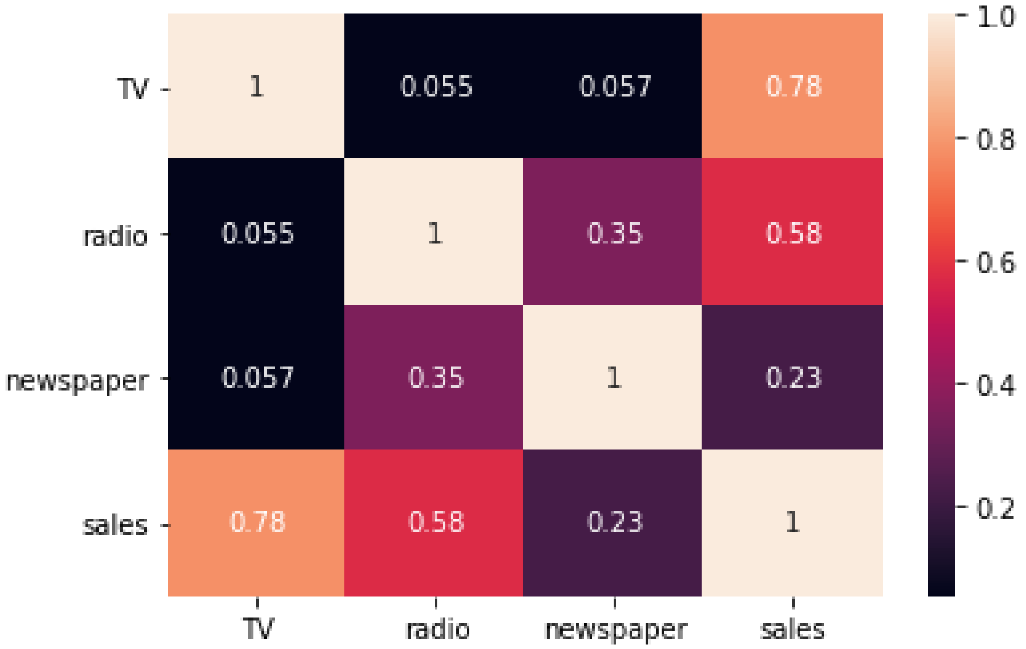
Then I checked the info() of df and be sure there is no null values and all the features are numeric:

```
[7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

I wanted to check the correlation of every column with each other with a heatmap:

sns.heatmap(df.corr(), annot=True);



Then I created X and y before moving to modelling.

X=df.drop(["sales"], axis=1)

y=df["sales"]

1.Linear Regression

I started with linear regression by importing modules, then split the X and y for training and testing:

1.Linear Regression

```
[12]: from sklearn.linear_model import LinearRegression

[13]: lm=LinearRegression()

[14]: 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)

[15]: lm.fit(X_train, y_train)

[15]: LinearRegression()
```

Then I define error metrics function as eval_metrics:

```
[11]: def eval_metrics(actual, pred):
          rmse = np.sqrt(mean_squared_error(actual, pred))
          mae = mean_absolute_error(actual, pred)
          mse = mean_squared_error(actual, pred)
          score = r2_score(actual, pred)
          return print("r2_score:", score, "\n","mae:", mae, "\n","mse:",mse, "\n","rmse:",rmse)
```

Check the errors:

```
[18]: y_pred=lm.predict(X_test)
```

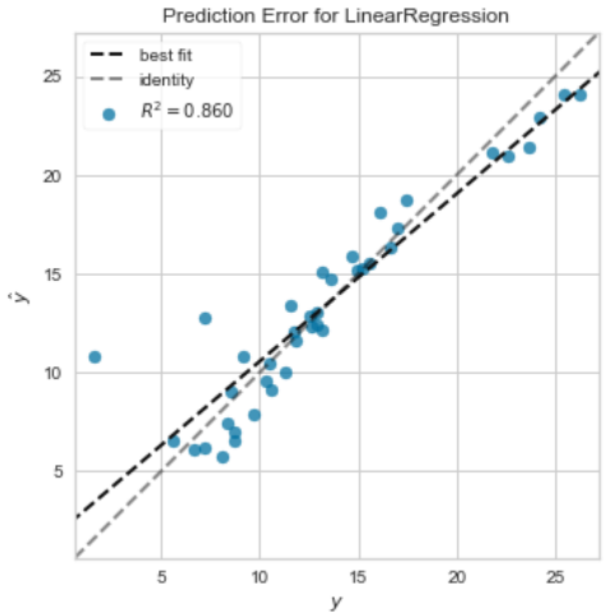
```
[19]: eval_metrics(y_test, y_pred)

r2_score: 0.8601145185017868
 mae: 1.361781350209028
 mse: 4.402118291449686
 rmse: 2.098122563495681
```

```
[20]: lm.score(X_test,y_test)

[20]: 0.8601145185017868
```

```
[21]: r2_score(y_test, y_pred)

[21]: 0.8601145185017868
```

And I used yellowbrick library and draw prediction error for LinearRegression graph:

```
[24]: from yellowbrick.regressor import PredictionError
      visualizer = PredictionError(lm)
      visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
      visualizer.score(X_test, y_test)  # Evaluate the model on the test data
      visualizer.show()
```



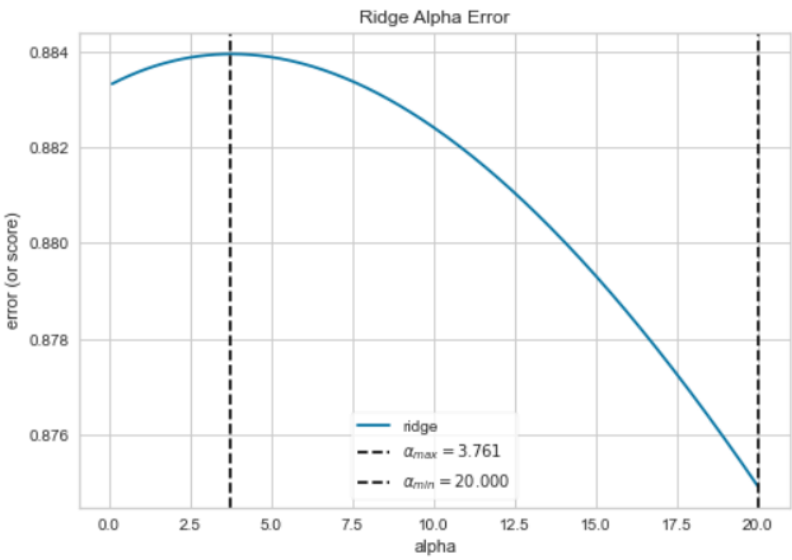
2.Ridge Regression:

[ ]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
```

30]:
```
ridge_model = Ridge(normalize=True)
```

After importing the modules I used the yellowbrick library to calculate the optimum alpha (lambda) values to use in ridge regression:

[54]:
```
#let's find the optimum alpha with yellowbrick
from yellowbrick.regressor import ManualAlphaSelection
# Create a list of alphas to cross-validate against
alpha_space = np.linspace(0.1, 20, 300)
# Instantiate the visualizer
visualizer = ManualAlphaSelection(
    Ridge(),
    alphas=alpha_space,
    cv=10)

visualizer.fit(X_train, y_train)
visualizer.show();
```



Here optimum alpha is 3.761. And I used this value in ridge regression and then calculated the model score:

[42]:
```
ridge_model.fit(X_train, y_train)
y_pred = ridge_model.predict(X_test)
```

[43]:
```
eval_metrics(y_test, y_pred)
```

```
r2_score: 0.8599750723184534
 mae: 1.3609514322231928
 mse: 4.406506585272249
 rmse: 2.099168069801046
```

[44]:
```
accuraries = cross_val_score(estimator=ridge_model, X=X_train, y=y_train, cv=10)
accuraries.mean()
```

[44]:
```
0.8836067293821358
```

3. Polynomial Regression

In the third part I used polynomial regression and hoped to reach a better score:

Polynomial Regression

```
[84]: # we will use not scaled (X original X)
      from sklearn.preprocessing import PolynomialFeatures
      polynomial_converter = PolynomialFeatures(degree=2,include_bias=False)
      poly_features = polynomial_converter.fit_transform(X)
      poly_features.shape

[84]: (200, 9)

[90]: X.shape

[90]: (200, 3)

[91]: X_train, X_test, y_train, y_test = train_test_split(poly_features, y, test_size=0.3, random_state=101)

[87]: model=LinearRegression()

[88]: model.fit(X_train,y_train)

[88]: LinearRegression()
```

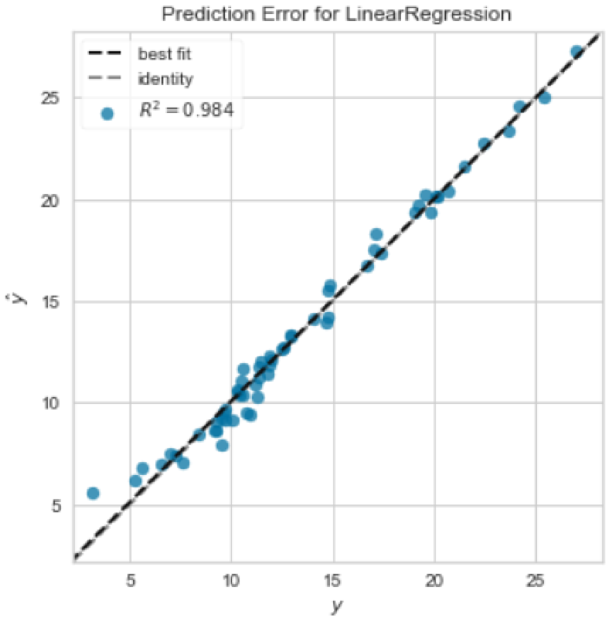
Then, I checked the errors and r^2 score:

```
[77]: y_pred=model.predict(X_test)

[78]: eval_metrics(y_test, y_pred)

      r2_score: 0.9843529333146787
       mae: 0.48967980448037
       mse: 0.4417505510403648
       rmse: 0.6646431757269196

[79]: accuraries = cross_val_score(estimator=model, X=X_train, y=y_train, cv=10)
      accuraries.mean()

[79]: 0.9814092804779175

[80]: accuraries

[80]: array([0.98721689, 0.99303246, 0.9905022 , 0.98369201, 0.98423858,
             0.93187129, 0.9907614 , 0.99330359, 0.96496016, 0.99451422])
```

We can see that r^2 score of polynomial regression is pretty higher than other models (linear and ridge).

Finally, I used yellowbrick library to draw the scatter plot of y_test values versus predicted y values:

```
[82]: from yellowbrick.regressor import PredictionError
visualizer = PredictionError(model)
visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.show();
```



Since my data set is pretty clean, does not have extreme outliers, and have just 3 features (independent), polynomial regression gives a successful result. The r^2 score is very high.