Problem 3

[K-Fold Cross Validation] In this problem, the goal is to use diabetes dataset from sklearn library and plot k-fold cross- validation scores against model complexity. Use polynomial regression, discussed in class to fit polynomial of degree k to the data. Search space for the degree of the polynomial can be taken to be $k \in [1, 10]$. Plot following curve: Cross Validation Score vs Degree of Polynomial Regression (Note: The plots may blow up for some model complexities. The goal is to infer this.) Report optimal choice of k based on cross val score() function in the sklearn library.

Importing necessary packages

```python
import math
import random
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.pipeline import Pipeline
from sklearn.datasets import load_diabetes
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures

num= 100
random.seed = 42
np.random.seed = 42
sns.set_style("darkgrid")

X, Y = load_diabetes(return_X_y = True, as_frame = True)
X = X[['age', 'sex', 'bmi', 'bp']]
print(X.head())
print(Y.head())
```

```
        age       sex       bmi        bp
0  0.038076  0.050680  0.061696  0.021872
1 -0.001882 -0.044642 -0.051474 -0.026328
2  0.085299  0.050680  0.044451 -0.005670
3 -0.089063 -0.044642 -0.011595 -0.036656
4  0.005383 -0.044642 -0.036385  0.021872
0    151.0
1     75.0
2    141.0
3    206.0
```

```
4    135.0
Name: target, dtype: float64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size =
0.8, random_state = 0)

# For degree 1
n = 1
pipel = Pipeline([('poly', PolynomialFeatures(n)),
                  ('scaler', StandardScaler()),
                  ('linea', LinearRegression())])
pipel.fit(X_train, Y_train)

# For degree 10
n = 10
pipel_10 = Pipeline([('poly', PolynomialFeatures(n)),
                  ('scaler', StandardScaler()),
                  ('linea', LinearRegression())])
pipel_10.fit(X_train, Y_train)
```

```
Pipeline(steps=[('poly', PolynomialFeatures(degree=10)),
                ('scaler', StandardScaler()), ('linea',
LinearRegression())])
```

```
print('In train set:')
print('The model trained with polynomial features of degree 1',
r2_score (Y_train, pipel.predict(X_train)))
print('The model trained with polynomial features of degree 10',
r2_score (Y_train, pipel_10.predict(X_train)))

print('In test set')
print('The model trained with polynomial features of degree 1',
r2_score (Y_test, pipel.predict(X_test)))
print('The model trained with polynomial features of degree 10',
r2_score(Y_test, pipel_10.predict(X_test)))
```

```
In train set:
The model trained with polynomial features of degree 1
0.4253556823737591
The model trained with polynomial features of degree 10 1.0
In test set
The model trained with polynomial features of degree 1
0.2740192519276704
The model trained with polynomial features of degree 10 -
270198.49137645063
```

Using validation data (standard method)

```
# Train validation split
X_train_new, X_val, Y_train_new, Y_val = train_test_split(X_train,
```

```
Y_train, train_size = 0.8, random_state = 0)

val_score = []
for i in range(1, 11):
  n = i
  pipelN = Pipeline([('poly', PolynomialFeatures(n)),
                     ('scaler', StandardScaler()),
                     ('linea', LinearRegression())])
  pipelN.fit(X_train_new, Y_train_new)

  Y_val_pred = pipelN.predict(X_val)
  r2 = r2_score(Y_val, Y_val_pred)
  val_score.append(r2)

for i in range(len(val_score)):
  print(f'for n = {i + 1} the r2 score is {val_score[i]}')
```

```
for n = 1 the r2 score is 0.3906511557827157
for n = 2 the r2 score is 0.29494011303425083
for n = 3 the r2 score is 0.08842108965674966
for n = 4 the r2 score is 0.0716492733682087
for n = 5 the r2 score is -1.9720872056870138
for n = 6 the r2 score is -384.32360841460945
for n = 7 the r2 score is -76791.71266746201
for n = 8 the r2 score is -654104351.0393577
for n = 9 the r2 score is -3353450.73752144
for n = 10 the r2 score is -287213.194140729
```

The most optimal value of degree of polynomial regression from R squared score is 1.

Using K-fold Validation data

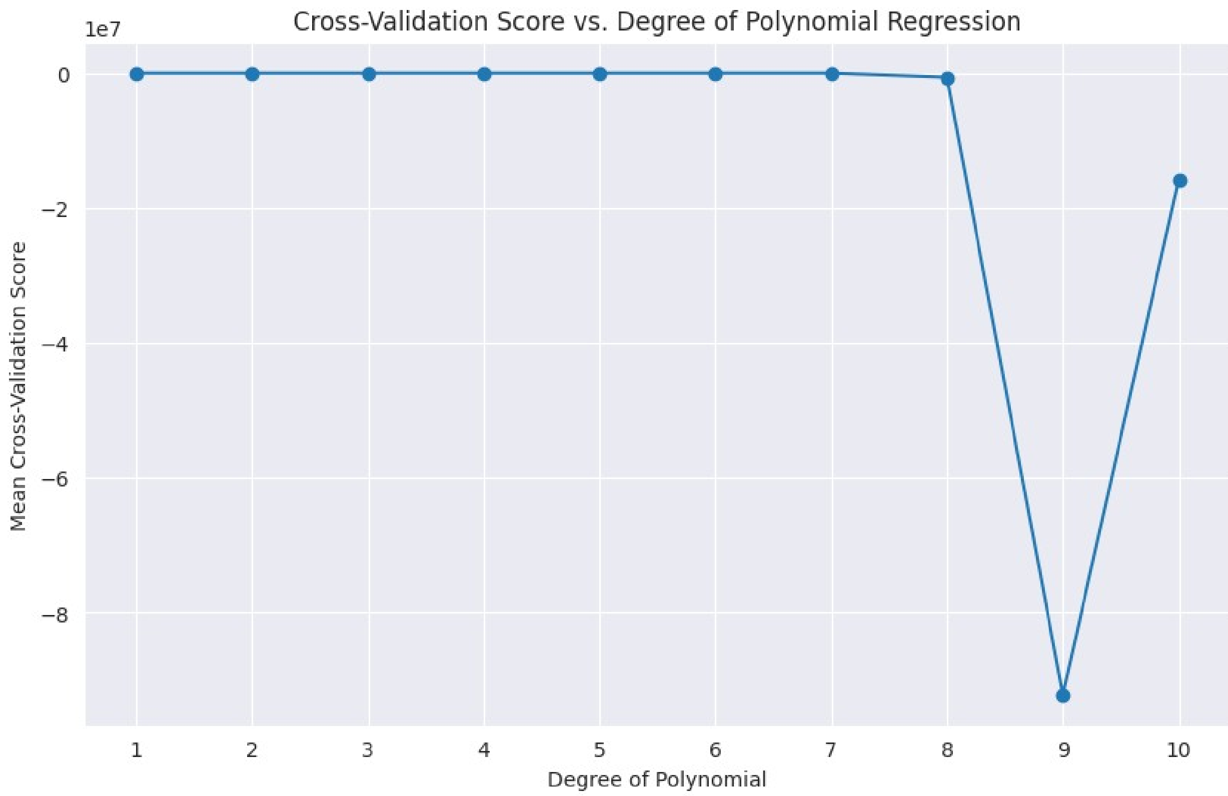
```
# K fold

val_score = []
for i in range(1, 11):
  n = i
  pipelN = Pipeline([('poly', PolynomialFeatures(n)),
                     ('scaler', StandardScaler()),
                     ('linea', LinearRegression())])

  c_val_score = cross_val_score(pipelN, X, Y, cv = 5)
  val_score.append(c_val_score)

for i in range(len(val_score)):
  print(f'for n = {i+1} the 5 fold cross validation scores are:
{val_score[i]}. \n')
```

```
for n = 1 the 5 fold cross validation scores are: [0.26674525
0.39888947 0.4044412  0.36174561 0.44085106].
```

```
for n = 2 the 5 fold cross validation scores are: [0.29185137
0.4148694  0.4115444  0.36153382 0.44716499].

for n = 3 the 5 fold cross validation scores are: [0.2536539
0.42396216 0.29426441 0.30035585 0.41031307].

for n = 4 the 5 fold cross validation scores are: [ 0.127432
0.33041057 -0.2376402   0.05233274 -0.52874947].

for n = 5 the 5 fold cross validation scores are: [-0.66584588
0.06143416 -0.68972109 -3.68048818 -1.74863488].

for n = 6 the 5 fold cross validation scores are: [  -4.65953005  -
11.36916347 -119.05432564 -396.81990538  -32.84833486].

for n = 7 the 5 fold cross validation scores are: [ -167.99117562  -
123.37035313  -585.17712374  -119.62323418
 -1809.26356993].

for n = 8 the 5 fold cross validation scores are: [  -11800.95458353
-70360.49535961 -1350682.27746277  -483454.11048919
 -1136808.29176504].

for n = 9 the 5 fold cross validation scores are: [-5.89376151e+06 -
1.50703663e+07 -1.08180865e+08 -5.83676863e+07
 -2.73054552e+08].

for n = 10 the 5 fold cross validation scores are: [  -761436.79129272
-959762.02534456 -66714408.2307476
  -3035130.07800055  -7336386.03230894].


# Since  val_score is a list of lists, where each inner list contains
cross-validation scores for a specific degree
# Convert it to a numpy array for easier manipulation
val_scores = np.array(val_score)

# Calculate the mean of the cross-validation scores for each degree
mean_scores = val_scores.mean(axis=1)

[print(f'for n = {i + 1} the mean score is {mean_scores[i]}') for i in
range(len(mean_scores))]

# Create an array of degrees for the x-axis
degrees = np.arange(1, 11)

# Plot the cross-validation scores vs. degree
plt.figure(figsize=(10, 6))
plt.plot(degrees, mean_scores, marker='o', linestyle='-')
plt.title('Cross-Validation Score vs. Degree of Polynomial
```

```
Regression')
plt.xlabel('Degree of Polynomial')
plt.ylabel('Mean Cross-Validation Score')
plt.grid(True)
plt.xticks(degrees)
plt.show()
```

```
for n = 1 the mean score is 0.3745345179227967
for n = 2 the mean score is 0.38539279680196864
for n = 3 the mean score is 0.3365098777114664
for n = 4 the mean score is -0.05124287343688407
for n = 5 the mean score is -1.34465117311117
for n = 6 the mean score is -112.95025188138845
for n = 7 the mean score is -561.0850913177359
for n = 8 the mean score is -610621.2259320281
for n = 9 the mean score is -92113446.39559889
for n = 10 the mean score is -15761424.631538872
```



Observation

Since we are using Linear Regression the default performance metrics used by cross_val_score is R^2, and we can see that it is highest for degree of polynomial regression = 2. Hence the most optimal value of degree of polynomial regression from k-fold cross validation is 2.