# Problem 2

[Akaike and Bayesian Information Criteria]  In this problem, the goal is to use Dataset 2 described in the tutorial notebook and plot AIC, BIC and AICc curves against model complexity. Use polynomial regression, discussed in class to fit polynomial of degree k to the data. Calculate AIC, BIC and AICc (described in the notebook). Search space for the degree of the polynomial can be taken to be $k \in [1, 30]$. Plot following 3 curves: AIC/BIC/AICc vs Degree of Polynomial Regression. Report optimal choice of k for each information criterion as well as corresponding test MSE.

## Importing necessary packages

```
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.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")

dataset_path = '/content/drive/MyDrive/sem 7/ID5055/Tutorial 
5/poly_reg2.csv'

# Function to load data and get train and test data
def load_data(path):
  data = pd.read_csv(path)
  arr = data.to_numpy().T

  return train_test_split(arr[0], arr[1], test_size = 0.2, 
random_state = 42, shuffle = True)

# Funtion for poly regression
def poly_regression(path, k = 2, print_values = True):
  x_train, x_test, y_train, y_test = load_data(path)

  poly_train = PolynomialFeatures(degree = k, include_bias = False)
```

```python
  poly_x_train = poly_train.fit_transform(x_train.reshape(-1, 1))
  poly_x_train = sklearn.preprocessing.normalize(poly_x_train)

  poly_test = PolynomialFeatures(degree = k, include_bias = False)
  poly_x_test = poly_test.fit_transform(x_test.reshape(-1, 1))
  poly_x_test = sklearn.preprocessing.normalize(poly_x_test)

  poly_model = LinearRegression()
  poly_model.fit(poly_x_train, y_train)
  y_pred_train = poly_model.predict(poly_x_train)
  y_pred_test = poly_model.predict(poly_x_test)

  mse_train = mean_squared_error(y_train, y_pred_train)
  mse_test = mean_squared_error(y_test, y_pred_test)

  # find AIC/BIC/AICc for the given model
  num = len(poly_model.coef_) + 1
  n = len(poly_x_train)
  AIC = n*math.log(mse_train) + 2*num
  BIC = n*math.log(mse_train) + math.log(n)*num
  AICc = n*math.log(mse_train) + 2*num + (2*num*(num + 1))/(n-num-1)

  if print_values:
    print('Degree:', k)
    print('AIC (TRAIN):', AIC)
    print('BIC (TRAIN):', BIC)
    print('AICc (TRAIN):', AICc)

  return AIC, BIC, AICc

# For dataset 2
aic, bic, aicc = poly_regression(dataset_path, k = 20, print_values =
False)
print('Degree:', 20)
print('AIC (TRAIN):', aic)
print('BIC (TRAIN):', bic)
print('AICc (TRAIN):', aicc)
```

```
Degree: 20
AIC (TRAIN): 48.038263062384345
BIC (TRAIN): 98.06082239053585
AICc (TRAIN): 63.969297545142965
```

```python
# Relation between degree of polynomial regression and MSE

def k_vs_mse(k, degree_of_polyfit):
    creterion_value_vs_k = {}

    for i in range(1, k):
        aic, bic, aicc = poly_regression(dataset_path, k=i,
print_values = False)
```

```
        creterion_value_vs_k[i] = [aic, bic, aicc]

    AIC_values = [goodness_val[0] for goodness_val in
creterion_value_vs_k.values()]
    BIC_values = [goodness_val[1] for goodness_val in
creterion_value_vs_k.values()]
    AICc_values = [goodness_val[2] for goodness_val in
creterion_value_vs_k.values()]
    degrees = list(creterion_value_vs_k.keys())

    [print(f'for k: {k} goodness values are
{creterion_value_vs_k[k]}') for k in creterion_value_vs_k.keys()]

    # Fit polynomial curves to the scatter points
    aic_coefficients = np.polyfit(degrees, AIC_values,
degree_of_polyfit)
    bic_coefficients = np.polyfit(degrees, BIC_values,
degree_of_polyfit)
    aicc_coefficients = np.polyfit(degrees, AICc_values,
degree_of_polyfit)

    # Create polynomial functions
    AIC_poly = np.poly1d(aic_coefficients)
    BIC_poly = np.poly1d(bic_coefficients)
    AICc_poly = np.poly1d(aicc_coefficients)

    x_values = np.linspace(min(degrees), max(degrees), 100)

    fig, ax = plt.subplots(1, 3, figsize=(12, 5))

    # Plot AIC vs Degree of Polynomial Regression (k) with the fitted
curve
    ax[0].scatter(degrees, AIC_values, label='AIC values')
    ax[0].plot(x_values, AIC_poly(x_values), label='Fitted
Polynomial', color='red')
    ax[0].set_ylabel('AIC values')
    ax[0].set_xlabel('Degree of Polynomial Regression')
    ax[0].set_title('AIC vs Degree of Polynomial Regression (k)')
    ax[0].legend()

    # Plot BIC vs Degree of Polynomial Regression (k) with the fitted
curve
    ax[1].scatter(degrees, BIC_values, label='BIC values')
    ax[1].plot(x_values, BIC_poly(x_values), label='Fitted
Polynomial', color='red')
    ax[1].set_ylabel('BIC values')
    ax[1].set_xlabel('Degree of Polynomial Regression')
    ax[1].set_title('BIC vs Degree of Polynomial Regression (k)')
    ax[1].legend()
```

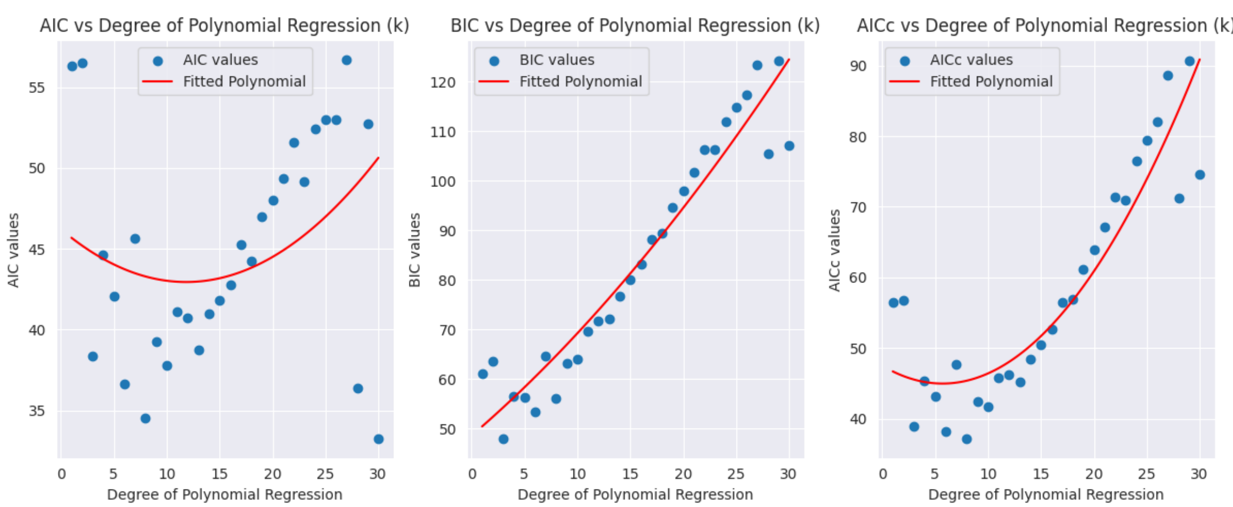
```
    # Plot AICc vs Degree of Polynomial Regression (k) with the fitted
curve
    ax[2].scatter(degrees, AICc_values, label='AICc values')
    ax[2].plot(x_values, AICc_poly(x_values), label='Fitted
Polynomial', color='red')
    ax[2].set_ylabel('AICc values')
    ax[2].set_xlabel('Degree of Polynomial Regression')
    ax[2].set_title('AICc vs Degree of Polynomial Regression (k)')
    ax[2].legend()

    plt.tight_layout()
    plt.show()

k_vs_mse(31, 2)
```

```
for k: 1 goodness values are [56.28362382382081, 61.04767709316857,
56.43946797966497]
for k: 2 goodness values are [56.51812450163737, 63.66420440565901,
56.83391397532158]
for k: 3 goodness values are [38.3878098016933, 47.915916340388826,
38.92114313502663]
for k: 4 goodness values are [44.62712756068876, 56.537260734058165,
45.43793837149958]
for k: 5 goodness values are [42.08120141077858, 56.37336121882187,
43.231886342285435]
for k: 6 goodness values are [36.63698390532576, 53.311170348042936,
38.19253946088132]
for k: 7 goodness values are [45.659692409938415, 64.71590548732947,
47.68786142402292]
for k: 8 goodness values are [34.56679358218486, 56.005033294249785,
37.13822215361343]
for k: 9 goodness values are [39.29190359595836, 63.112169942697165,
42.48030939305981]
for k: 10 goodness values are [37.77210322999954, 63.974396211412234,
41.65445617117601]
for k: 11 goodness values are [41.0852575422678, 69.66957715835437,
45.74197396017824]
for k: 12 goodness values are [40.74156084991042, 71.70790710067088,
46.25671236506194]
for k: 13 goodness values are [38.78259428429718, 72.13096716973152,
45.24413274583564]
for k: 14 goodness values are [40.97571002699764, 76.70610954710587,
48.47571002699764]
for k: 15 goodness values are [41.83932885848351, 79.95175501326561,
50.474249493404145]
for k: 16 goodness values are [42.7787344100135, 83.27318719946948,
52.649702151948986]
for k: 17 goodness values are [45.26470890633885, 88.14118833046871,
56.47782366043721]
for k: 18 goodness values are [44.230240568869505, 89.48874662767325,
```

```
56.89690723553617]
for k: 19 goodness values are [46.970361556346944, 94.61089424982457,
61.20764969194016]
for k: 20 goodness values are [48.038263062384345, 98.06082239053585,
63.969297545142965]
for k: 21 goodness values are [49.376824877100525, 101.78141083992591,
67.1312108420128]
for k: 22 goodness values are [51.60257130383719, 106.38918390133645,
71.3168570181229]
for k: 23 goodness values are [49.17082608685432, 106.33946531902747,
70.98900790503613]
for k: 24 goodness values are [52.39015573207503, 111.94082159892206,
76.4642298061491]
for k: 25 goodness values are [53.0005194623922, 114.9332119639131,
79.49108550012805]
for k: 26 goodness values are [53.01598689484152, 117.33070603103631,
82.0929099717646]
for k: 27 goodness values are [56.71335935393057, 123.41010512479924,
88.55649660883253]
for k: 28 goodness values are [36.39423634460403, 105.47300875014659,
71.19423634460404]
for k: 29 goodness values are [52.70767364048206, 124.1684726806985,
90.66685731395145]
for k: 30 goodness values are [33.24252935430757, 107.0853550291979,
74.5758626876409]
```




# Observations

For degree of polynomial regression = 8 we have the lowest [AIC, BIC, AICc] combinaton, which is [34.56679358218486, 56.005033294249785, 37.13822215361343].

Hence the optimal model has degree of polynomial regression = 8.

If we see individually

1. **AIC**: the optimal model has degree of polynomial regression is 30 with score = 33.24252935430757, and MSE value for test data for k = 30 is 7.985318788837407.

2. **BIC**: the optimal model has degree of polynomial regression is 3 with score = 47.915916340388826 and MSE value for test data for k = 3 is 2.0479225453996497.

3. **AICc**: the optimal model has degree of polynomial regression is 8 with score = 37.13822215361343 and MSE value for test data for k = 8 is 2.1086299148178536.