Task 1

Ordinary least squares Linear Regression.
LinearRegression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

The problem solved by the sklearn library to find the coefficients is called linear least square problem where we try to find the x that minimizes the norm of the residual error $||Ax - b||^2$ Where A is the $m \times n$ matrix and b is m dimensional. This problem is solved by solving the equation :

$$A^T A\, x = Ab$$

This system of equations are called the normal equations for the linear least-squares problem.

Task 2

Degree	Errors	Bias	Variance	Irreducible Error
1	1.056923e+06	834.474907	51994.979434	0.000000e+00
2	1.032068e+06	828.628868	74130.146184	-1.164153e-10
3	8.031461e+04	204.481440	73346.365204	0.000000e+00
4	1.140164e+05	232.277952	107179.517277	0.000000e+00
5	1.224180e+05	239.151253	115914.097150	-1.455192e-11
6	1.533543e+05	271.692786	147325.200686	-2.910383e-11
7	1.642627e+05	285.782822	157063.767120	0.000000e+00
8	2.203152e+05	318.454659	213022.052628	0.000000e+00
9	2.353528e+05	329.634029	227702.669551	2.910383e-11
10	2.496894e+05	339.306069	241819.010150	0.000000e+00
11	2.649836e+05	320.416548	257690.063671	-5.820766e-11
12	2.809375e+05	327.077252	265156.021762	0.000000e+00
13	2.656636e+05	311.619413	257513.151257	0.000000e+00
14	2.805525e+05	306.761840	256905.083620	5.820766e-11
15	3.067994e+05	333.742429	256141.564292	0.000000e+00
16	3.189851e+05	327.718321	264109.291248	0.000000e+00
17	3.687000e+05	373.120213	260602.261152	-5.820766e-11
18	3.806801e+05	370.002731	269905.090753	0.000000e+00
19	4.638462e+05	427.851017	270287.767448	5.820766e-11
20	4.754401e+05	426.034298	279183.177945	1.164153e-10

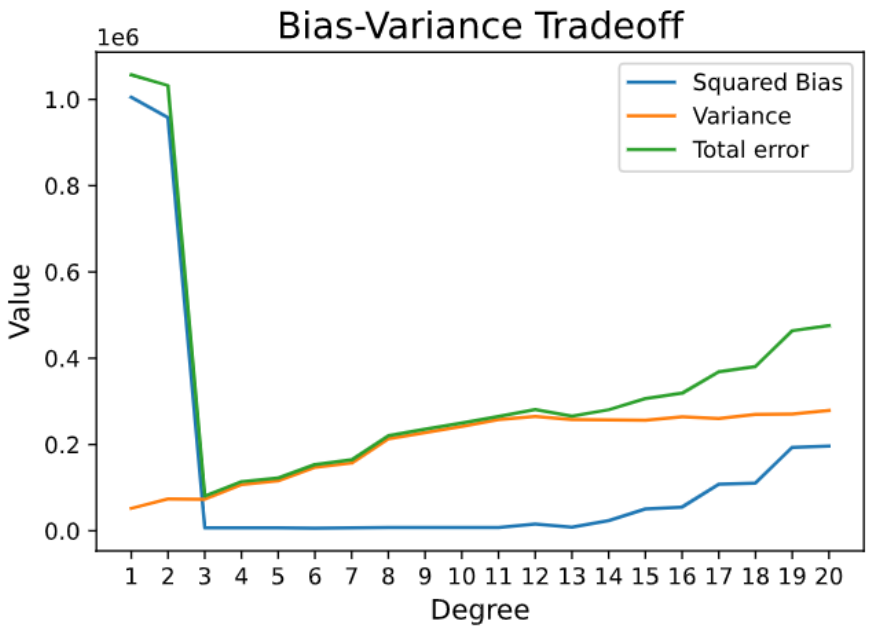
We conclude that degree 3 polynomial is ideal for this dataset.

At the degrees less than 3 we had a very high bias and MSE. It can be observed all error terms attain a minimum at degree 3. For higher degrees the error again increases, which is due to multicollinearity of the high degree polynomial features. Multicollinearity makes the model statistically very unstable and even small changes to the training set can lead to high variation of the coefficients in the model. As a result the errors including the bias can be seen rising in higher degrees.

Task 3

Irreducible Error value remains very close to 0 (infact in the range of 1e-10 which could even be floating point precision error) for every degree, this indicates that the data has no noise. The irreducible error does not change with the class of models because the irreducible error is a property of the data given and is independent of the model used. No matter what model we use either a perfect model or not so perfect, we can not reduce/alter this error.

Task 4



For degree 1 and 2 the model is underfitting. For degree 4 onwards the variance increases significantly indicating overfitting. Degree 12 onwards bias also increases indicating underfitting. Since all error metrics were at minimum at degree 3, the dataset has been taken from a function that can be most approximated to a 3 degree polynomial.

Degree	Behaviour
1	Underfitting
2	Underfitting
3	
4	Overfitting
5	Overfitting

6	Overfitting
7	Overfitting
8	Overfitting
9	Overfitting
10	Overfitting
11	Overfitting
12	Underfitting due to multicollinearity
13	Underfitting due to multicollinearity
14	Underfitting due to multicollinearity
15	Underfitting due to multicollinearity
16	Underfitting due to multicollinearity
17	Underfitting due to multicollinearity
18	Underfitting due to multicollinearity
19	Underfitting due to multicollinearity
20	Underfitting due to multicollinearity

Code Explanation:

In order to fit a polynomial to the training data, we can use linear regression by transforming the features appropriately. For x, if we create the features x^2, x^3, x^4 and so on, then we can use linear regression to get a polynomial. We have done this using the sklearn.preprocessing.PolynomialFeatures class. We can now fit a

regression model to the data. Now in order to make predictions from the data, we convert the test data into polynomial features also.

We train each model (i.e. polynomial of a particular degree) 10 times, once on each partition. The performance metrics of the model are computed by calculating its bias and variance on each data point (in the test set) across the 10 model instances. The mean bias (or variance) of the model is then the mean of the bias (or variance) values calculated for each data point (in the test set).

The outer loop iterates over the 20 different models (i.e. polynomials of different degrees), while the inner loop iterates over the 10 different realizations of the model in consideration. Here, the model is trained on one of the partitions of the training set, and then tested on the test set. The code uses vectorization instead of for loops to improve performance.

The predictions of the model are then stored. After all 10 instances have been trained and tested for a particular model, we find the average difference in predicted and actual values for each data point, averaged across the 10 instances, which gives the bias of the model for each data point. Similarly, variance is also calculated. The bias (or variance) of the model is then the average of the bias (or variance) for each data point.

For each degree, regression fit plots have been shown (on the test dataset). Because we have 10 instances of each model, we have just used the first one (since each is trained on a random partition of the dataset).