1. Introduction and Problem Statement

The goal of the Take home is to fit several Regression model on a given Training data set, and to get the best model to predict on an unseen Test dataset. The Train dataset contains a 2D independent feature set (X1, X2) and 200 realizations of an unknown function (Y(X1,X2)) on these independent features. We will calculate the mean and variance of these random observations, and then build separate models that can predict the these mean and variance values. Finally, we will use them to predict the unknown mean and variance from observations in the Test dataset.

2. EDA

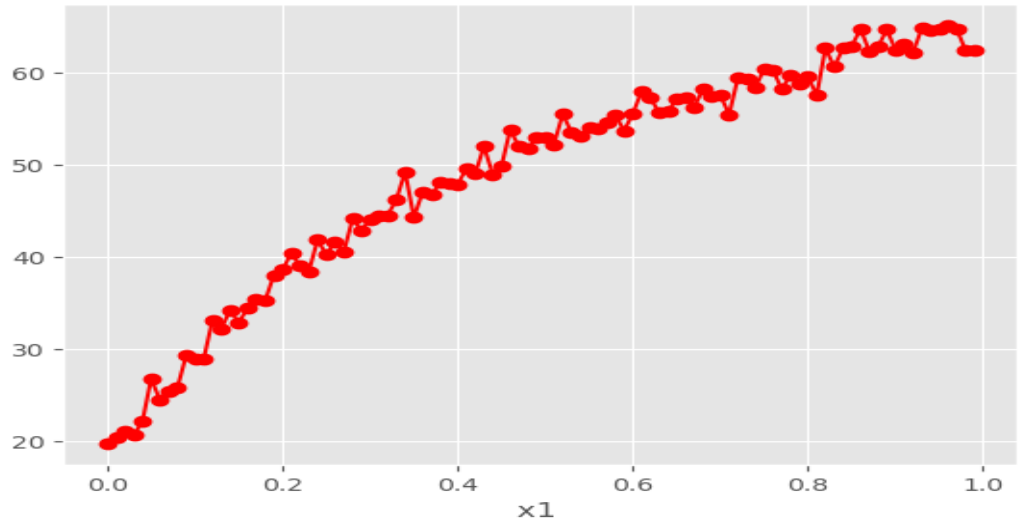
The Train Dataset contains **10000** observations with **202** features. The first two are the independent features (X1 and X2) while the remaining **200** are the random observations. The X1 and X2 values range from **0 to .99** with a mean of **.49**. For the mean value of the 200 random observations, a min and max of **4.59 and 69.5** were obtained with a mean of **44.4**. Similarly for the variance, a min and max of **1.56 and 416.49** were obtained with a mean of **234.11**.

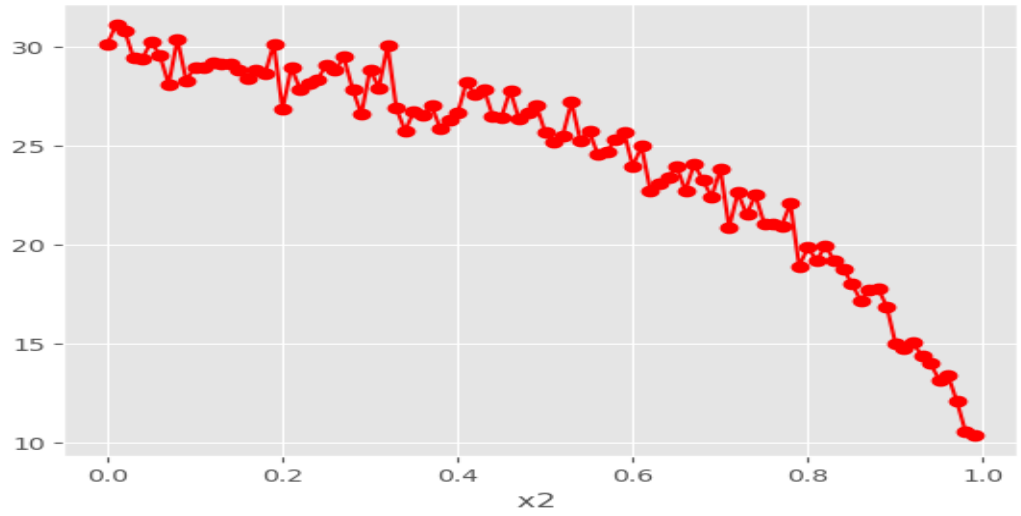
And the test Dataset contains 2500 observations for which we must predict the unseen mean and variance.

Some visuals are given below:

1. **Mean distribution**:

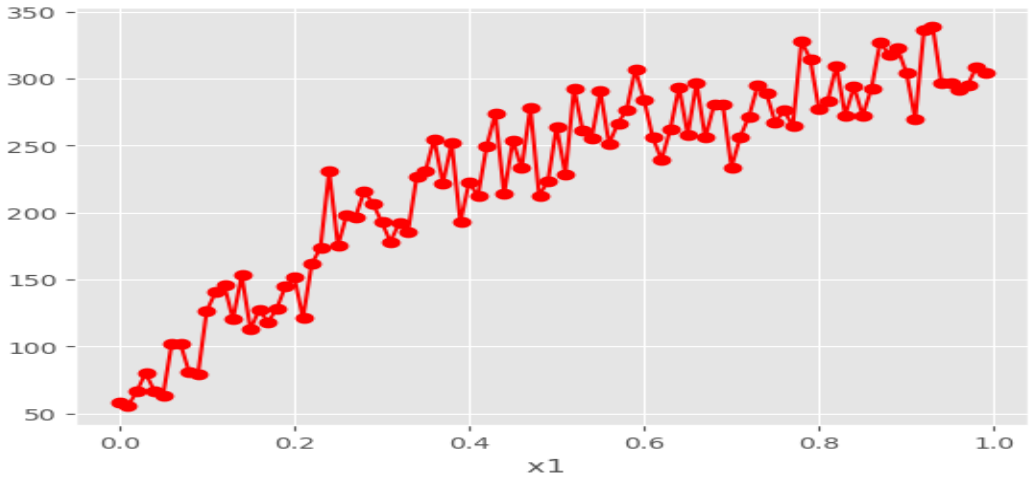
If we hold X2 static (as an example value=.1) and look at the variation of the mean on different X1 values, we see the below. We can see that the mean values are increasing for higher values of X1, with some dips in between. The image right below it shows the reverse. We hold X1 constant (value =.1) and look at y values for different values of X2. Here we see the opposite behavior, which is a dip for increasing values of X2, with some rises.

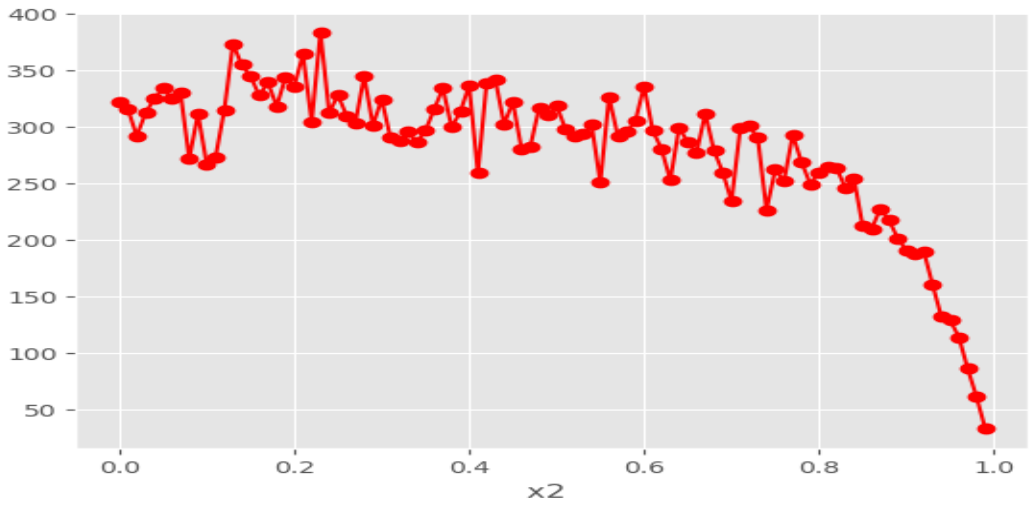




2. **Variance distribution**:

Next, we look at similar plots for Variance. If we hold X2 static (value=.7) and look at the variation of the variance values on different X1's, we see the below. We can see that the variance values are increasing for higher values of X1, with some dips. The image right below it shows the reverse. We hold X1 constant (value =.7) and look at y values for different values of X2. Here we see the opposite behavior, which is a dip for increasing values of X2, with some rises and it tapers towards the end.





3. Methodologies Applied

Below are the different models that were fitted on the Training dataset.

Also, instead of Accuracy, which is normally used for classification, we would be using the **R2 score** to get to the best model.

An R-Squared value shows how well the model predicts the outcome of the dependent variable. R-Squared values range from 0 to 1. An R-Squared value of 0 means that the model explains or predicts 0% of the relationship between the dependent and independent variables. A value of 1 indicates that the model predicts 100% of the relationship, and a value of 0.5 indicates that the model predicts 50%, and so on.

- Tree Based: **Random Forest**

  A meta estimator that fits several decision tree regressors on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
- Boosting
  - ***XGBoost**, which stands for Extreme Gradient Boosting, is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library.*
  - ***AdaBoost** regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.*
  - ***GradientBoostingRegressor**: This estimator builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.*
- Polynomial Features
  - *Generate a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. Later this is fit to a **Linear Regression Model***

- Generalized additive Models

*The **GAM** framework is based on an appealing and simple mental model: Relationships between the individual predictors and the dependent variable follow smooth patterns that can be linear or nonlinear. Mathematically speaking, GAM is an additive modeling technique where the impact of the predictive variables is captured through smooth functions which—depending on the underlying patterns in the data—can be nonlinear.*

- Support Vector Regression

***Support vector regression** (SVR) is a type of support vector machine (SVM) that is used for regression tasks. It tries to find a function that best predicts the continuous output value for a given input value.*

SVR can use both linear and non-linear kernels. A linear kernel is a simple dot product between two input vectors, while a non-linear kernel is a more complex function that can capture more intricate patterns in the data. The choice of kernel depends on the data's characteristics and the task's complexity.

- *K Nearest Neighbors*

  *Regression based on **k-nearest neighbors**. The target is predicted by local interpolation of the targets associated with the nearest neighbors in the training set.*

4. Analysis and Results

First, using CV we got the optimal hyperparameters for each model. And then using Monte Carlo Cross Validation, we obtained the optimal R2 values, and the mean squared difference from the actual mean / variance values.

1. Model runs to fit and predict **Mean**

Model	*Optimal hyperparameters via CV*	*Train R2 Score based on Monte Carlo run*	*Mean Square Error from Actual Mean*
GAM	NA	.9828	3.17
GBR	{'subsample': 0.5, 'n_estimators': 100, 'max_depth': 8, 'loss': 'huber', 'learning_rate': 0.1, 'ccp_alpha': 0.001}	**.9929**	**1.05**
XGBoost	{'n_estimators': 250, 'max_depth': 4, 'learning_rate': 0.0656530612244898, 'colsample_bytree': 1}	**.9931**	**1.09**
KNN	{'n_neighbors': 10}	**.9928**	**1.08**
Random Forest	{'n_estimators': 100, 'min_samples_split': 20, 'max_depth': 50, 'criterion': 'squared_error', 'ccp_alpha': 0.0030204081632653063}	**.9927**	**1.11**
SVR	{'kernel': 'rbf', 'gamma': 1, 'epsilon': 0.015142857142857145, 'degree': 4, 'C': 5}	**.9923**	**1.38**
AdaBoost	{'n_estimators': 500, 'loss': 'linear', 'learning_rate': 3.4934934934934936}	.9884	2.05
PolynomialFeatures with Linear Regression	Degree = 4	.9502	9.3

We can see that GBR, XGBoost, K Nearest Neighbors and Random Forest and SVR gave > **.99 R2 values**. All of them had lower Mean Squared Error from the actual Mean values. Then based on t-test performed on the MC validation runs, I was able to get that **KNN** performed the best, and hence I will be using that to build the final model to evaluate the mean.

2. Model runs to fit and predict Variance

Model	*Optimal hyperparameters via CV on Train data*	*Train R2 Score based on Monte Carlo run*	*Mean Square Error from Actual Variance*
GAM	NA	**.9091**	**700.43**
GBR	{'subsample': 0.5, 'n_estimators': 250, 'max_depth': 3, 'loss': 'squared_error', 'learning_rate': 0.1, 'ccp_alpha': 0.07979591836734694}	**.9279**	**517.07**
XGBoost	{'n_estimators': 250, 'max_depth': 3, 'learning_rate': 0.05757142857142858, 'colsample_bytree': 1}	**.9286**	**521.97**
KNN	{'n_neighbors': 16}	**.9246**	**515.31**
Random Forest	{'n_estimators': 250, 'min_samples_split': 20, 'max_depth': 10, 'criterion': 'squared_error', 'ccp_alpha': 0.06161224489795919}	**.9270**	**465.2**
SVR	{'kernel': 'rbf', 'gamma': 1, 'epsilon': 0.015142857142857145, 'degree': 4, 'C': 5}	.8744	931.7
AdaBoost	{'n_estimators': 250, 'loss': 'linear', 'learning_rate': 0.970970970970971}	**.9051**	**711.9**

PolynomialFeatures with Linear Regression	Degree = 4	.715	2214.7

We can see that GAM, AdaBoost, GBR, XGBoost, K Nearest Neighbors and Random Forest and SVR gave > **.90 R2 values**. All of them had lower Mean Squared Error than the actual variance values. Then based on t-test on the MC validation runs, I was able to get that **Random Forest** performed the better, hence I will be using that to build the final model to evaluate the variance.

Therefore, I used the optimal KNN Model to calculate the mean from the X1, X2 values of the Test set, and used the optimal Random Forest model to calculate the variance.

5. Conclusion

This was a great final project to identify models for regression and then choose the best one for an unseen dataset. There is no guarantee that the final model chosen would provide the best result on the Test set (for all that we know, one of the least-better models might have performed better), but it is this unknown aspect that makes Machine Learning much more flexible and robust against different variant scenarios.