8.02.2024

Report for Prediction of Canadian Households

Author: Alperen Demirci - Mail: alperendemirci65@gmail.com

Introduction

What is Regression ?

Regression is a **statistical technique used to model the relationship** between a dependent variable and one or more independent variables. The goal of regression analysis is to understand **how the dependent variable changes** as the independent variables change.We can think of it as a mathematical equation that uses predictor variables to predict a target variable.Regression helps us to understand the data better as it helps to visualize the relation between variables.

Tricky part of the regression is, it assumes that there exists a linear or polynomial relation between variables.From the nature of regression, it mostly has a **bias towards its kind**. What I mean is if we are to use Linear Regression, then model assumes that **there exists a linear relation** between the variables. This bias may or may not help us on our project due to the ***bias - variance tradeoff***.

We use regression for mostly for **predicting a continuous target**. Let's say that we want to build a model that predicts the **price of a product, average temperature** of the next day. Regression can be used on varying fields like Economics, Engineering, Medicine.

There exists a special kind of regression called Logistic Regression which doesn't predict continuous variables. It is used for classification and it's actually one of the powerful and simple models used on the field. The main reason why is it used for classification lies on the function that it uses: Logistic Function. It generates a probability using the features of entry. Then if the probability is lower than the threshold, it's classified as class 0 (vice versa).

Equation for Multiple Linear Regression: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$

Y: Dependent variable

$X_1, X_2, \ldots, X_p$: Independent variables

β_0: Intercept term

$\beta_1, \beta_2, \ldots, \beta_p$: Coefficients associated with the independent variables

ε: Error term

Types of Data for Regression:

You can use regression analysis with various types of data:

- Continuous Data (Mostly Used): Data that can take any value within a range, such as height, weight, temperature, etc.
- Categorical Data: Data that falls into categories, such as gender, type of product, etc. (for logistic regression).
- Time-Series Data: Data collected over a period of time, such as stock prices, sales figures, etc. (for time series regression).
- Cross-Sectional Data: Data collected at a single point in time, such as survey data, experimental data, etc.

Description of the Models

Multiple Linear Regression

We've given the general formula for this model. Let's review it.

Equation for Multiple Linear Regression: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon$

Y: Dependent variable
$X_1, X_2, \ldots, X_p$: Independent variables
β_0: Intercept term
$\beta_1, \beta_2, \ldots, \beta_p$: Coefficients associated with the independent variables
ε: Error term

Multiple Linear Regression is valuable in situations where there are multiple factors influencing an outcome and understanding the relationships between these factors is important for prediction, explanation, or hypothesis testing. However, it's important to ensure that the assumptions of multiple regression are met (biassed).

Image 2.A: Visualization of Multiple Linear Regression

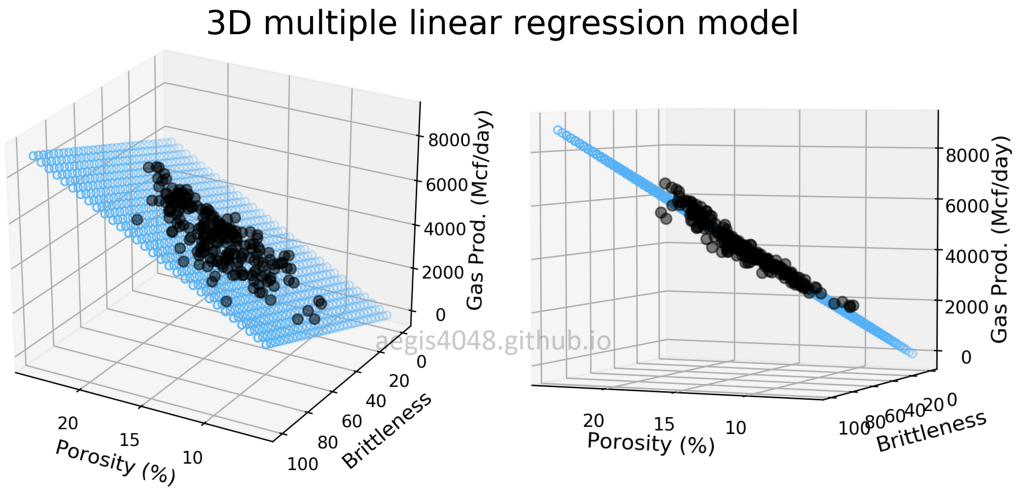

Image Source: https://aegis4048.github.io/mutiple_linear_regression_and_visualization_in_python

If we are to give example, we can use Multiple Linear Regression to predict a students' grade by processing the amount of time studied, midterm results, attention span time(in terms of hours). Naturally we assume that there is a linear relation between the predictors and the target variable. In this case our bias has a positive impact on our prediction.

To sum up, this model can be really effective when it comes to analyzing relations between features. It's not only used for predicting a variable, it is used in Explanatory Data Analysis too. MLR makes us easier to see the correlation between features. MLR is fast, used in various scenarios and very easy to implement.

Support Vector Regression

Support Vector Regression (SVR) works like drawing a line through a bunch of points on a graph. It wants to make sure most points are close to the line, but some can be a bit farther away. SVR picks out special points called support points that are really important for deciding where the line goes. It tries to find the best line that fits most points while still leaving some room for error. SVR can handle tricky patterns in the data by using a trick to see things in a different way. It tries to be just right—not too simple and not too complicated—so it can make good guesses about new points based on where they are compared to the line. So, SVR helps find a good line that fits the data well without being too strict or too loose, making it useful for understanding relationships in data.

Image 2.B: Visualization of regression and classification using Support Vectors.

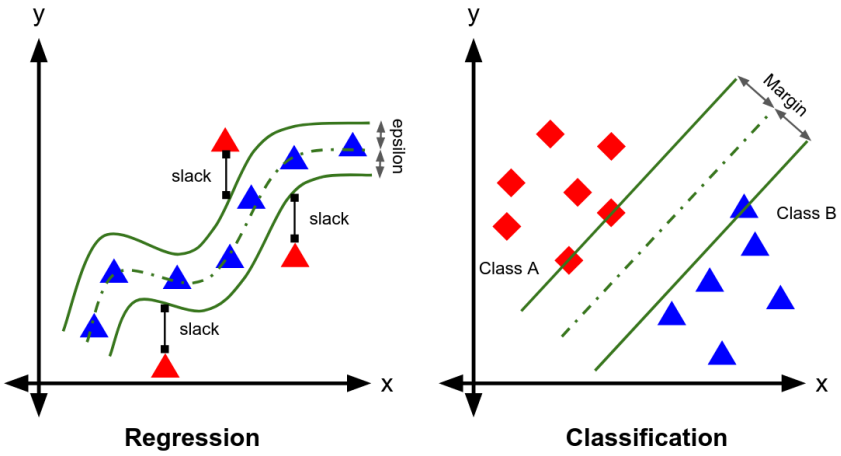

Image Source: https://medium.com/it-paragon/support-vector-machine-regression-cf65348b6345
Support Vector Regression (SVR) is a powerful tool in various domains where **traditional regression techniques might not perform well**, especially when dealing with **nonlinear and complex relationships** in the data. SVR is mostly used in **Financial Forecasting,Anomaly Detection,Bioinformatics.**

In summary, SVR is a versatile regression technique that can be applied in diverse fields where accurate prediction and modeling of **complex relationships** are required. It is particularly useful when dealing with nonlinear data and **high-dimensional feature spaces**.

KNN Regression

KNN is the simplest algorithm that we'll talk about in this paper. K-Nearest Neighbors (KNN) Regression predicts the value of a new data point by averaging the values of its closest neighbors in the feature space. For instance, if you want to predict the price of a house based on its size, KNN looks at the sizes of K nearby houses and averages their prices to make its prediction. It's like asking your neighbors for advice on how much your house might be worth. KNN Regression is straightforward and relies on the idea that similar data points should have similar outcomes, making it a simple yet effective method for prediction tasks.

*Fun note: In Turkish there is phrase called "Bana arkadaşını söyle, sana kim olduğunu söyleyeyim." which describes the behavior of KNN in a wise and simple way :) .

Image 2.C: Visualization of regression using KNN.

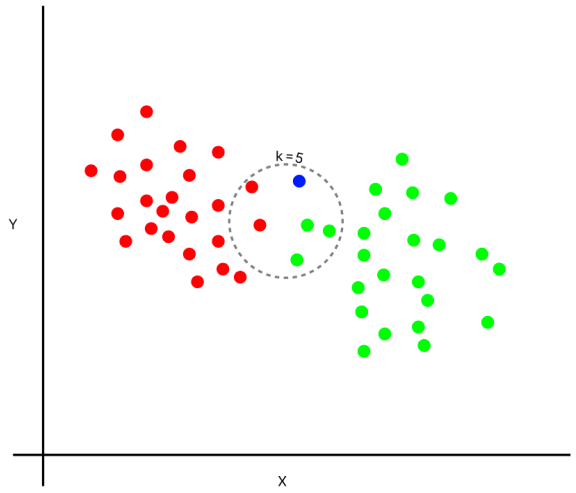

Image Source : https://pub.towardsai.net/understanding-k-nearest-neighbors-a-simple-approach-to-classification-and-regression-e4b30b37f151

KNN Regression is effective when the relationship between the independent and dependent variables is **nonlinear**, as it makes predictions based on the **proximity of data points**. It's a **naive** algorithm which doesn't need any training. It works well with **small to medium sized data** because there exist a **computational cost** which increases due to the size of data. KNN Regression works efficient for **localized predictions** since it's the main logic behind it.

In short, KNN Regression is a flexible method useful in different areas. It's great for figuring out relationships in data that aren't **straight** lines and making predictions based on **nearby points**. It's faster thanks to the simplicity of model. But, it's essential to think about how long it takes to calculate and how well it handles big sets of data. Also, picking the right **number of neighbors** (K) is key for the best results.

Random Forest Regressor

A Random Forest Regressor operates by creating an **ensemble** of **decision trees**, where each tree is trained on a **random subset** of the available training data and features. Through this ensemble approach, the algorithm **combines the predictions of multiple trees** to generate a final prediction for regression tasks. During the training phase, each decision tree is constructed using a process called **recursive binary splitting**, where the algorithm identifies optimal features and thresholds to partition the data into smaller subsets.

By averaging the predictions of all trees in the ensemble, the Random Forest Regressor provides a more **accurate** and **stable** prediction while minimizing the risk of overfitting. The random picking of data and features makes sure that each tree in the model learns different things about the data, which helps make the model **strong** and **dependable**.

Risks of using Random Forest Regressor comes from the basis of it: Decision Trees. They are very likely to overfit. However in Random Forest Regression it avoids overfitting by setting a **splitting threshold** which means the node of a tree won't split if the split has lower size then the threshold (empirically threshold is selected near 20). **Randomness** factor in training the trees with different part of the data contributes to the avidness of overfitting.

Image 2.D: Visualization of Random Forest Regression

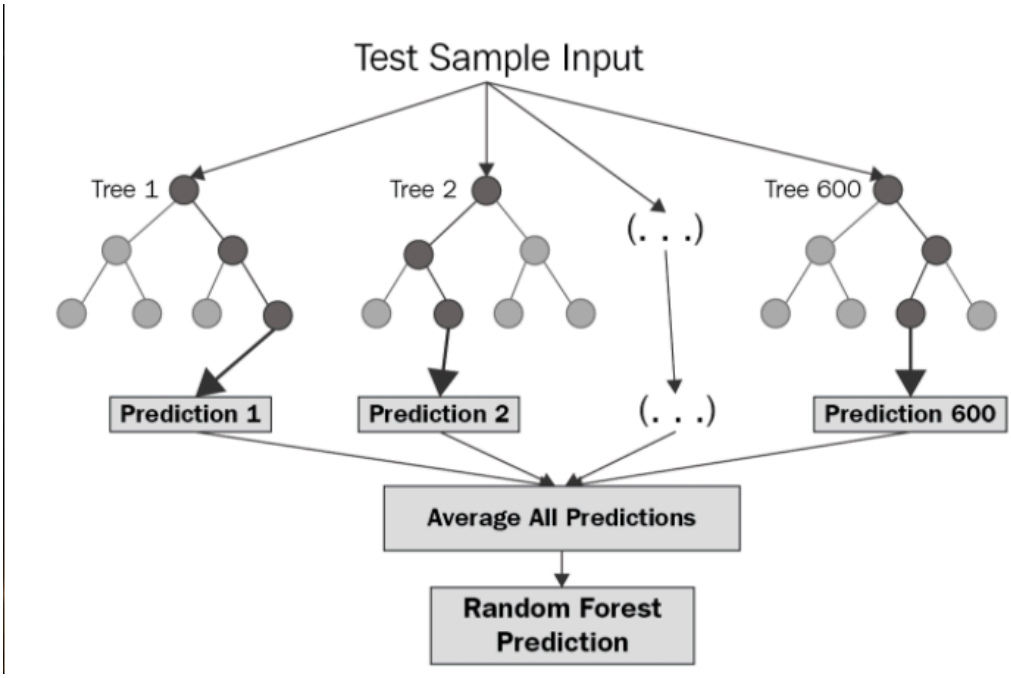

Image Source: https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

In simple terms, Random Forest Regressor is great for many types of datasets. It works well with data that **has lots of features**, different types of variables, and **complex** relationships between them. It can handle **noisy** data and **missing values** without any issues. Also, we can interpret the importance of variables by checking the nodes in different trees. Random Forest Regressor is flexible, strong, and gives results that are easy to understand.

Gradient Boosting Regression

Gradient Boosting is a clever tool in machine learning used for both classification and regression tasks. It brings together the predictions of several weak learners, usually simple decision trees, to form a robust predictive model. The main idea is to improve the model's accuracy by iteratively adding new models that focuses the errors of the old ones.

Image 2.E: Visualization of Gradient Boosting Regression

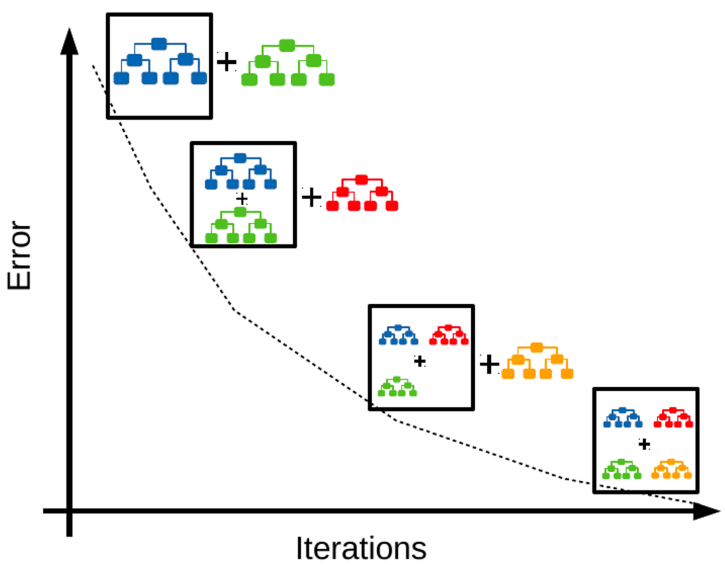

Image Source: https://www.almabetter.com/bytes/

tutorials/data-science/gradient-boosting
Here's how Gradient Boosting works:

1. **Start Simple**: We kick off with a basic model, often a decision tree, which predicts based on input features but might not be super accurate.

2. **Learn from Errors**: We examine where this basic model fails by comparing its predictions to the actual results.

3. **Correct Mistakes**: Next, we build a new model that focuses on fixing these errors. This model tries to predict the differences between the actual results and the predictions of the basic model.

4. **Blend Predictions**: We mix the predictions of the basic model with those of the new corrector model to improve overall accuracy.

5. **Repeat**: We repeat this process, each time concentrating on the errors made by the combined models and constructing new correctors to rectify them.

6. **When to Stop**: We keep adding correctors until we're content with the accuracy of our predictions or until we hit a set limit.

7. **Predict**: Finally, we utilize the combined models to predict outcomes on new data.

So, in essence, Gradient Boosting is about learning from mistakes and continuously retuning the model to improve predictions. It's like assembling a team where each member corrects the mistakes of the previous one, resulting in more accurate predictions overall. It's similar to the Random Forest Classifier since they both come from the same method called ***'ensemble'.*** The main difference is that in Gradient Boosting, new weak learners aim on the faults of the current model whilst Random Forest just *randomly* creates bunch of decision trees. They both have their advantages on their own. Let's compare these two briefly.

Comparison of Random Forest and Gradient Boosting

1. **Training Process:**
   - Gradient Boosting: Builds the ensemble sequentially by adding models that correct the errors of the previous ones. Each new model focuses on reducing the residual errors of the combined ensemble.
   - Random Forests: Constructs multiple decision trees independently and combines their predictions through averaging (for regression) or voting (for classification).

3. **Optimization Objective:**
   - Gradient Boosting: Minimizes a loss function by iteratively improving the model's predictions. It focuses on reducing the residuals or errors between the predicted and actual values.
   - Random Forests: Focuses on reducing variance by averaging multiple decision trees. Each tree is constructed independently without any consideration for the errors made by other trees.

4. **Handling of Outliers:**
   - Gradient Boosting: Can be sensitive to outliers since it tries to minimize the errors of the previous models. Outliers may have a greater impact on subsequent models.
   - Random Forests: Robust to outliers since the final prediction is based on the average or majority vote of many trees, which tend to cancel out the effect of outliers.

5. **Parallelization:**
   - Gradient Boosting: Typically harder to parallelize since each new model depends on the previous ones.
   - Random Forests: Easily parallelizable since each tree can be trained independently of the others.

Multiple Layer Perceptron Regression (Neural Network)

An MLP (Multi-Layer Perceptron) regressor is a type of neural network used for regression tasks. It has layers of neurons: input, hidden, and output. Each neuron processes input data using weighted connections and activation functions. During training, it adjusts weights to minimize the difference between predicted and actual outputs. Once trained, it predicts continuous numerical values for new data. In essence, it learns to map input features to continuous output values through iterative adjustment of its internal parameters. We can say that it's more complicated version of Linear Regression with different intuitions.

Image 2.F: Visualization of MLP compared to Linear Regression.

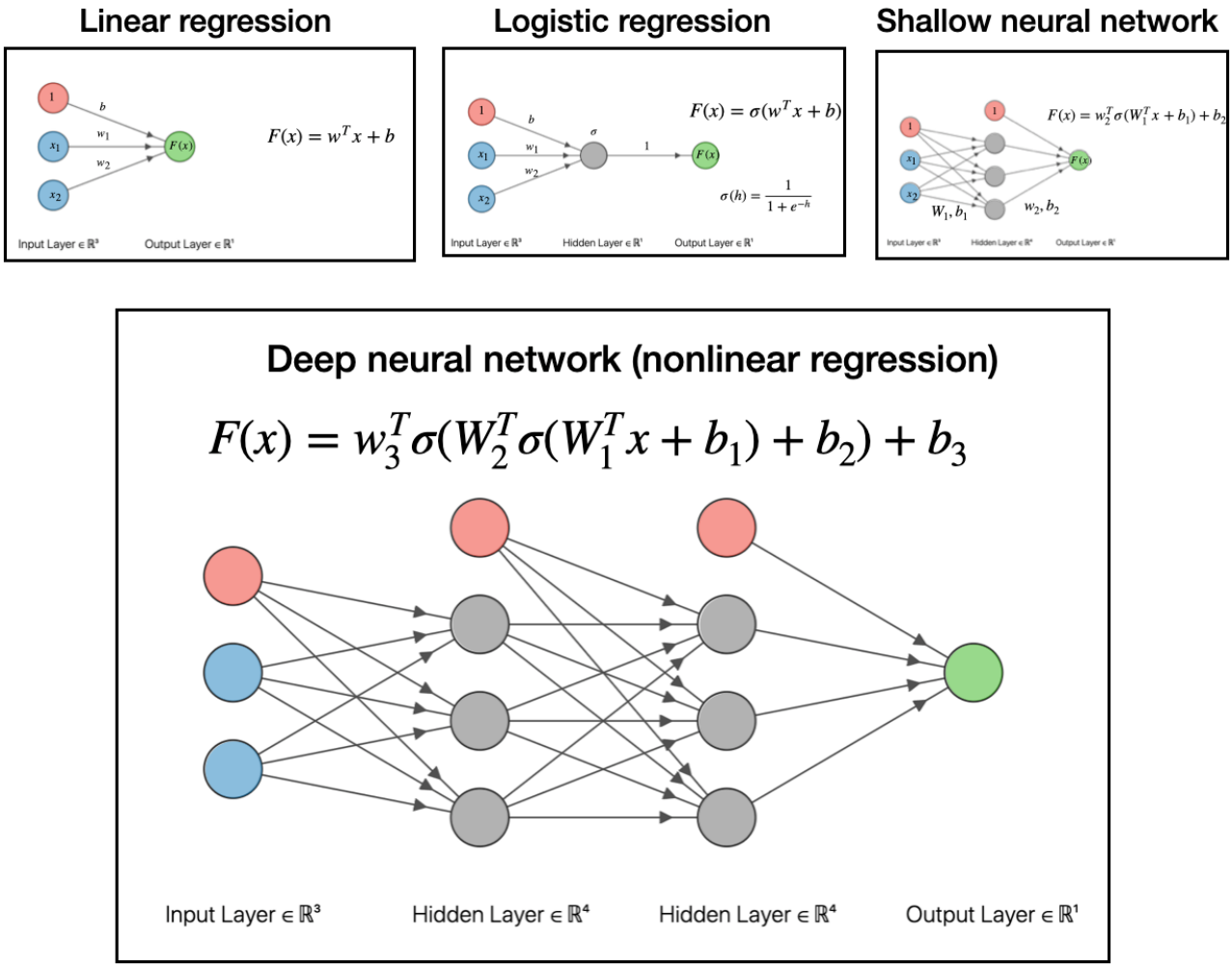

Image Source: https://joshuagoings.com/2020/05/05/neural-network/

Multi-Layer Perceptron (MLP) models offer versatile advantages in machine learning tasks. They excel in capturing **complex**, **non-linear** relationships between input features and output targets. MLPs can handle **various data types and distributions**, automatically learn relevant features, and scale effectively with increased complexity and dataset sizes. Their **universal function approximation capability** enables them to make regression on the vast majority of continuous functions. Additionally, MLPs can generalize well to unseen data, make use of **parallel computing** architectures for efficient training, and r**equire minimal feature engineering**. Negative sides of it are **cost to train** and hard to interpret results from the model.Overall, these attributes make MLPs a powerful and widely applicable tool across different domains and machine learning applications.

Explained Steps of the Data Processing

Data Processing

This part includes several subtitles given in order.

1. Checking Missing Values: Found **no** missing values.
2. Checking Unique Values: Dropped ‘Address’ column.
3. Outlier Detection and Handling: Applied **log transformation** to the data
4. Encoding with One Hot Encoding: Encoded categorical variables.
5. Normalization using Z-Score

Explanatory Data Analysis (EDA)

This part includes several subtitles given in order.

1. Correlation Matrix: Dropped columns ‘Population’ and ‘Province_Quebec’.
2. Clustering Analysis: Couldn’t cluster the data.
3. Plots of features - target: Visualizing patterns between features.
4. Distributions of features

Model Training

This part includes several subtitles given in order.

1. Train Test Split: Split the data with **80:20** ratio **without stratifying**.
2. Model Building
    1. K Fold Cross Validation: Selected **K as 5.**
    2. Hyper parameter Tuning with Grid Search

3. Multiple Linear Regression
4. Support Vector Regression
5. KNN Regression
6. Random Forest Regression
7. Gradient Boosting Regression
8. MLP (Neural Network) Regression

Model Evaluation

This part includes several subtitles given in order.

1. R2,MSE,MAE score comparison
2. Model Comparison

Conclusion

This part includes several subtitles given in order.

1. Which model to pick?
2. Comments

Model Comparison

Model	Mean Absolute Error (MAE)	R-Squared	Mean Squared Error (MSE)
Multiple Linear Regression	0.45609	0.58482	0.40182
kNN Regression (k=25)	0.38981	0.68328	0.30653
Random Forest Regression	0.37841	0.69851	0.29178
Support Vector Regression	0.37145	0.69589	0.29432
Neural Network Regression	0.38350	0.69533	0.29486
Gradient Regression	0.37667	0.70025	0.29009

• Note: Results may differ slightly due to the restart of the python kernel.

Conclusion and Evaluation

This is not a hard question for this project since the data speaks for itself. When we check the table above we can say that the **ensemble** principled methods **(Random Forest and Gradient Regression)** made a good job on this data. Others models aside, **Neural Network** intuition has fitted well compared to other algorithms.

The main reason behind this is the patterns in the features. For instance if we check the distribution of Median_Family_Income, you can see that there exist a **non linear but observable** pattern. The target (besides 2 or 3 variables) has **non linear relation** with features. In these situations **ensemble techniques and Neural Networks** have advantage over the other models.

When we compare **Neural Networks** and SVR, we can see that their metrics are really close. If we compare their training time, most of the time Neural Networks cost more to train. In this project SVR has taken more time since we didn't put same amount of parameters while Grid Searching. However, for this project we can say that their performances are similar and fine.

If we talk about other models, **KNN** has done a great job due to its' simplicity. If we compare KNN to **SVR**, we can see that SVR has taken time to train, more complex and it carries risk of **overfitting**. Again there exist a dilemma: simplicity - precision. Since SVR is a more advanced algorithm than KNN and it also is better given the results; we would probably choose SVR over KNN.

Unfortunately the **Linear Regressio**n has given the **worst** result among models. Reason for this is the reason why Neural Networks and ensemble methods work well: **linearity**. In the beginning, I've mentioned about regression models having a **bias** towards its' kind. Here, Linear Regression has found the linear relations perfectly. Although, it also needs to cover the **non linear/complex** relations too. Unless it handles that, it won't be useful as other models.

• Winner: **Gradient Regression** (Ensemble)

- Note: In the simplicity part, I didn't choose the simplest model. I've compared Gradient Regression over MLP since there doesn't exist a huge difference in evaluation metrics. Otherwise, I wouldn't mention an opportunity to choose between two models, we should simply choose the least erroneous one.

Thanks for reading!
Please mail me if you've noticed any mistake or have a suggestion!