# **Group Project - Kaggle Challenge**

# **Regression with a Tabular Paris Housing Price Dataset**

In this code, we are working with a dataset containing information about houses. Our goal is to predict the median value of houses based on different features such as MedInc, HouseAge, AveRooms, AveBedrms, Population, AveOccup, Latitude, Longitude. This data can be accessed through the following link:

<https://www.kaggle.com/competitions/playground-series-s3e6/data?select=train.csv>

We started by loading the data and exploring it through visualizations like histograms, box plots, and kernel density estimation plots.

To start, we focused on the "MedInc" feature from the dataset. we plotted several visualizations, including a bar plot, box plot, violin plot, and kernel density estimation (KDE) plot. Through these plots, we observed that the data is skewed, indicating that it is not evenly distributed. Additionally, we noticed the presence of outliers, which are data points that significantly deviate from the majority of the data.

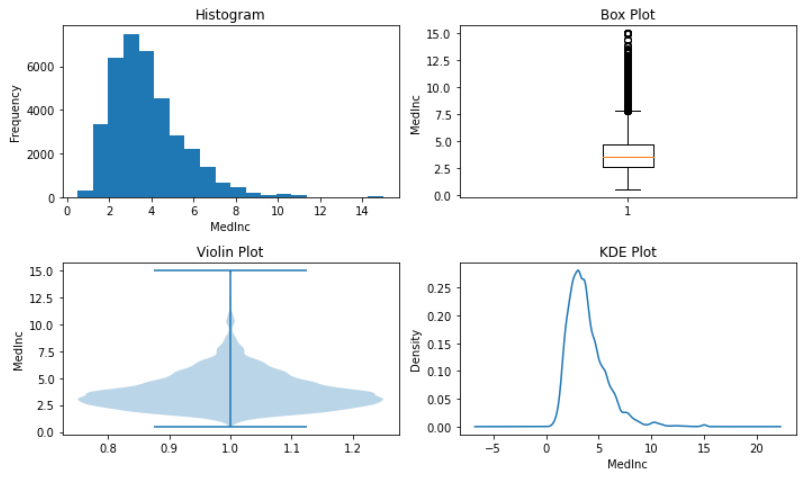


Figure 1 - Visualizations used for analysis

Next, we utilized the latitude and longitude coordinates to create a heatmap, representing the geographical distribution of the data points. The heatmap displayed a concentration of data points in the California area, indicating that a majority of the data corresponds to locations within California.

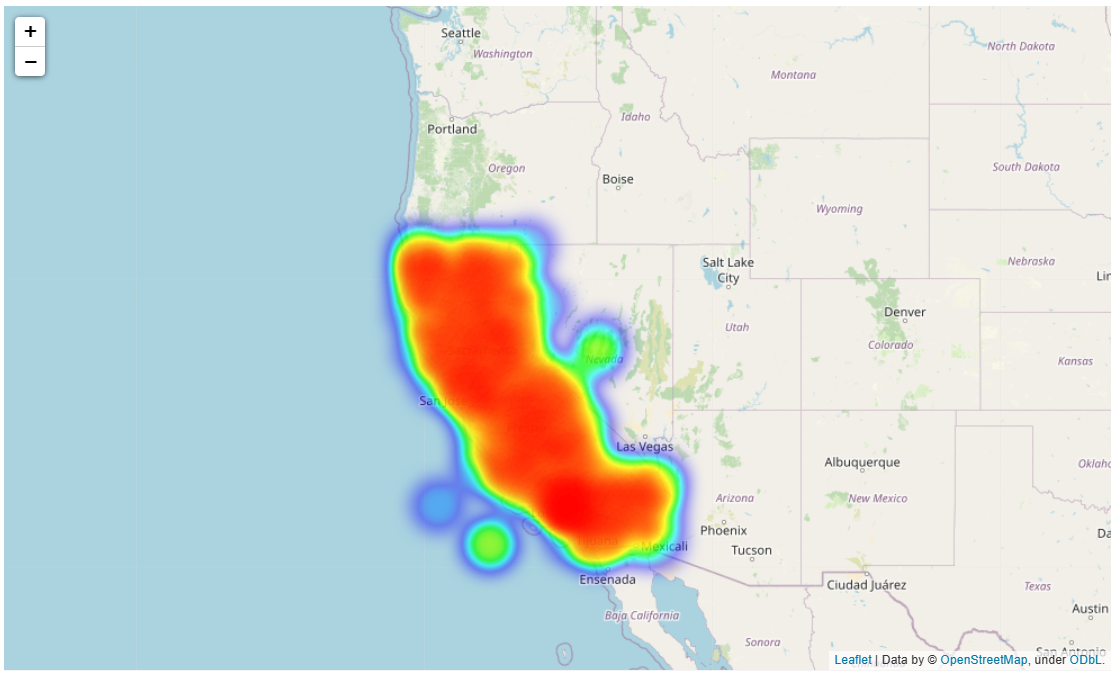


Figure 2 - Map visualisation showing the location based on coordinates

We also checked the correlation between the features using a heatmap. The correlation heatmap revealed a strong correlation of 0.7 between the AveRooms and MedInc variables. As a result, we made the decision to drop the AveRooms variable from the dataset.

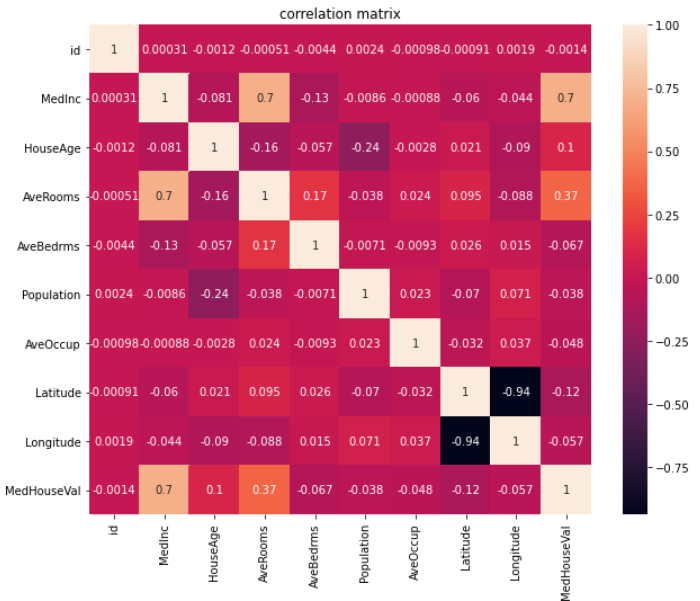


Figure 3 - correlation heatmap

The model architecture we used in this assignment is a random forest regressor, which is an ensemble learning method that combines multiple decision trees to make predictions. The random forest regressor is known for its ability to handle complex relationships between features and target variables (Dubey, 2020). To train the model, we divided our data into a training set and a validation set. We used the training set to fit the model and adjust its parameters. The training scheme we employed involved iterating over different values of the hyperparameter "n\_estimators," which represents the number of decision trees in the random forest. We tested multiple values for "n\_estimators" ranging from 10 to 100. For each value of "n\_estimators," we trained a separate random forest regressor model and evaluated its performance using mean squared error (MSE). The MSE measures the average squared difference between the predicted and actual median house values. We computed the MSE for both the training set and the validation set.

We stored the MSE values for each model. To determine the optimal number of trees, we plotted the MSE values against the number of estimators and identified the elbow point. Looking at the training errors, we can observe a decreasing trend as the number of estimators increases. This suggests that adding more trees to the random forest leads to better performance on the training data. However, it's important to note that the training error tends to decrease with increasing model complexity, and it may not always be a reliable indicator of the model's generalization ability. The validation errors provide insights into how well the model performs on unseen data. For smaller values of "n\_estimators" (10 to 30), the validation error decreases, indicating improved performance. However, beyond a certain point (around 30), the validation error starts to stabilize or slightly increase as the number of estimators further increases. This suggests that the model's performance on the validation data plateaus, indicating a trade-off between model complexity and generalization. Based on the results, the optimal value for "n\_estimators" is identified as 90, which corresponds to the lowest validation error of 0.053. This optimal model achieves a training error of 0.049, indicating a good fit to the training data.

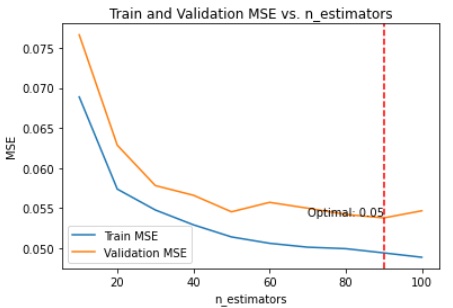


Figure 4 - Train and validation error

Using the 90 estimators, we built a final random forest regression model and made predictions on the test data. Finally, we prepared a submission file for the predictions. Overall, this code helps us analyse the dataset, find patterns in the data, and build a model to predict house prices accurately.

## **References**

Dubey, B. (2020, January 8). Random Forest Regression. Medium. Retrieved from https://medium.com/@bhartendudubey/random-forest-regression-d751df81cc54

Rakhra, M. et al. (2021) "WITHDRAWN: Crop Price Prediction Using Random Forest and Decision Tree Regression:-A Review", Materials Today: Proceedings. doi: 10.1016/j.matpr.2021.03.261.

Smith, P., Ganesh, S. and Liu, P. (2013) "A comparison of random forest regression and multiple linear regression for prediction in neuroscience", Journal of Neuroscience Methods, 220(1), pp. 85-91. doi: 10.1016/j.jneumeth.2013.08.024.

Zhang, J. et al. (2019) "Modelling uniaxial compressive strength of lightweight self-compacting concrete using random forest regression", Construction and Building Materials, 210, pp. 713-719. doi: 10.1016/j.conbuildmat.2019.03.189.