Università degli Studi di Milano

Data Science for Economics

Bayesian Analysis Project

Bayesian Regression Analysis
for Predicting Salaries in Data Jobs

by
Giordano Vitale
14310A

and
Emile Rahal
10744A

submitted to
Prof. Dr. Luca Rossini

May 8, 2024

Contents

1 Introduction **1**

2 Data Description **2**

2.1 Dataset 2

2.2 Preprocessing Techniques 2

3 Linear Regression **4**

3.1 The Bayesian Framework 4

3.2 Bayesian Linear Regression - Normal Priors 5

3.2.1 Results 6

3.3 Bayesian Linear Regression - Student-T Priors 7

3.3.1 Results 7

3.4 Hyperparameter Tuning 8

4 Ridge Regression **9**

4.1 Results 10

5 Regression Tree **11**

5.1 Bayesian Regression Tree 11

5.1.1 Perpendicular Regression Tree 11

6 Conclusions **13**

6.1 General Evaluations 13

6.2 Potential Improvements 14

7 References **15**

1 Introduction

The project's goal is to use Bayesian approach to perform a detailed regression analysis aimed at predicting American workers' salaries in the Data Science industry and related sectors. Using Bayesian techniques, we aim to provide accurate salary estimates while simultaneously identifying the most important drivers and providing insights to bolster the career path of a Data professional. Specifically, we aim to assess and compare our models based on metrics such as Mean Absolute Error (MAE) Root Mean Squared Error (RMSE), and the Bayes Information Criterion (BIC) to ensure accurate salary predictions.

For this, we experiment with 3 different regression techniques and propose a total of 5 models to investigate which approaches yield the most desirable results. During the study, we experimented with different prior beliefs and we also leveraged Bayesian optimization during hyperparameter tuning. To promote an easy comparison between all the models, we will propose a table of metrics that we applied on every model.

In addition to our research findings, we developed a user-friendly web-based application designed to allow users to predict their salary based on their input data (that serves as *out-of-sample predictions*).

The project is implemented in Python and the entire code can be found in our GitHub repository (see References).

2 Data Description

2.1 Dataset

The dataset considered for this project consists of over 700 observations and 41 feature variables - including binary, numerical, and categorical domains - and the target variable, the *Salary*. The columns collectively provide a comprehensive overview of the job postings, company details, and associated requirements. Out of the 41 predictors, we have excluded certain columns that were deemed irrelevant to our analysis or redundant for our purposes (for instance *Competitors*).

The outcome of the preprocessing steps will be a cleaned data set consisting of 14 columns and 306 observations. It has been divided into training and test sets, with a proportion of 80% and 20% respectively.

2.2 Preprocessing Techniques

Preprocessing data is a crucial step, as it enhances important features that improve the models' performance. Moreover, it plays a significant role in reducing overfitting.

While there were no null values, there was a weighty proportion of duplicates. We decided to remove them to reduce the dependence of our models on these specific observations, enhancing their generalization abilities.

All the preprocessing steps discussed below have been performed solely based on the training set features, without incorporating information from the test set, in order to prevent data leakage. This approach helps mitigate the risk of overfitting and increases the external validity of our models.

Variables Encoding

Since most algorithms require numerical inputs, encoding categorical variables is essential as it allows us to convert categorical data into a numerical format that algorithms can understand and process effectively.

For categorical columns with a meaningful order, such as *Revenue*, we opted for Ordinal Encoding. This method assigns integers to the categories based on their inherent order, preserving the relationship between different levels of the variable.

$$\text{OrdinalEncoder}(\textit{Revenue}) = \begin{cases} 0 & \text{if } \textit{Revenue} = 1 \text{ to } 5 \text{ million (\$)} \\ \dots & \\ 10 & \text{if } \textit{Revenue} = 10+ \text{ billion (\$)} \end{cases}$$

For categorical columns without a natural order among their values, such as *Job Title*, we deployed Target Encoding. This technique replaces each category with the average target value - in our case the Salary - for that category. As an example, the value *Company - Private* within the variable *Type of ownership* has been encoded with 93.53, which represents the average salary of the employees working at private companies.

Scaling

By scaling the data we prevent features with larger scales from dominating those with smaller scales during model training, ensuring fair and effective learning across all features. We thus applied MinMax scaling to our data to ensure that all features have a consistent scale. This scaling method rescales the data to a specific range, typically between 0 and 1, preserving the relative relationships between different features.

Feature Engineering

Many columns of the original dataset captured the employee's skills via binary variables. To shorten and simplify the data to work with, we performed a grouping of similar skills into broader, coherent categories. While we applied this approach to several variables, we hereby provide one example only for brevity reasons.

The variable *PyTorch* is defined as:

$$\text{PyTorch} = \begin{cases} 1 & \text{if employee knows PyTorch,} \\ 0 & \text{otherwise} \end{cases}$$

and its new broader category is the following:

$$\text{MachineLearning}_i = \text{Keras}_i + \text{SciKit} - \text{Learn}_i + \text{PyTorch}_i + \text{TensorFlow}_i$$

Outliers Detection

During our exploratory analysis, we employed the IQR method and we detected one severe outlier for the target column *Salary.* We opted to remove it from the dataset. This decision was motivated by the significant impact the outliers had on the statistical properties of the column, which affected the modeling efforts.

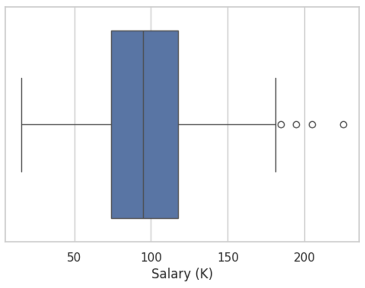


Figure 2.1: Boxplot for the target variable

3 Linear Regression

The first family of models that we propose belongs to Linear Regression. This choice is motivated by the simplicity and clarity these models benefit from. We propose two implementations of the Bayesian approach, one with Normal priors and the second with Student-T priors, with the aim to investigate whether differences in the priors can yield significant changes in the posterior distributions, hence potentially affecting the overall performances. We then performed a comparison between one Bayesian version and its frequentist-based companion.

3.1 The Bayesian Framework

The Bayesian linear regression model can be represented by the following formula:

$$p(\beta|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\beta)\cdot p(\beta)}{p(\mathbf{y}|\mathbf{X})}$$

Where:

- $p(\beta|\mathbf{X},\mathbf{y})$ represents the posterior distribution of the parameters (β) given the data ($\mathbf{X}$ and $\mathbf{y}$).
- $p(\mathbf{y}|\mathbf{X},\beta)$ represents the likelihood of the observed data given the parameters.
- $p(\beta)$ represents the prior distribution of the parameters, reflecting our initial beliefs about their values prior to data observation.
- $p(\mathbf{y}|\mathbf{X})$ represents the marginal likelihood or evidence, serving as a normalization constant.

By combining the likelihood and prior distributions using Bayes' theorem, we compute the posterior distribution of the parameters given the data. This allows us to not only make predictions but also quantify the uncertainty associated with those predictions.

The linear regression model can be represented in matrix notation as:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon$$

Where:

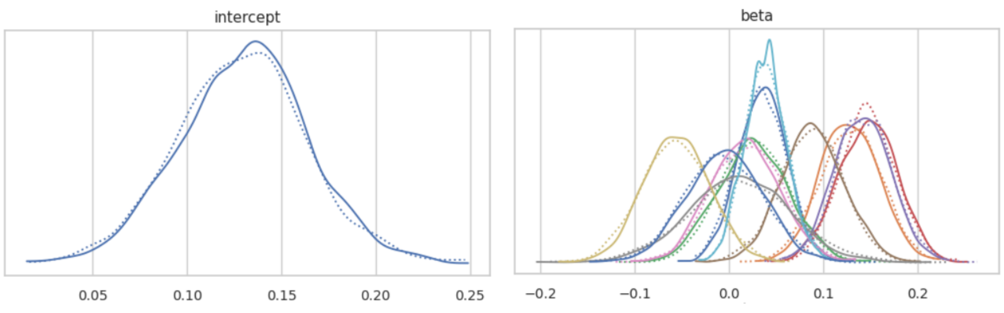
- $\mathbf{y}$ represents the target variable.
- $\mathbf{X}$ represents the feature matrix.
- β represents the vector of coefficients (including the intercept).
- ε represents the vector of errors.

3.2 Bayesian Linear Regression - Normal Priors

The first Linear Model we implemented is a linear model whose coefficients have been estimated through the application of the Bayes theorem. As anticipated before, the prior distributions for the intercept (*Int*) and the variables' coefficients (β_i) have the following statistical property:

$$Int, \beta_i \sim N(\mu{=}0, \sigma{=}10) \quad i = 1, \ldots, 11$$

This specification reflects the *non-informative prior* idea, for we assumed there was limited available information, thus allowing the parameters to vary quite widely. This approach represents a cautious and conservative strategy, avoiding the imposition of strong prior beliefs that may bias the analysis.



(a) Intercept posterior (b) Coefficients' posteriors

Figure 3.1

The posterior distributions of the intercept and coefficients, shown in Figure 3.1, are then obtained through NUTS (No-U-Turn Sampler), a form of Markov Chain Monte Carlo (MCMC) sampling: this process is an adaptive version of the HMC (Hamiltonian Monte Carlo) which automatically tunes the step size parameter during sampling to efficiently explore the posterior distribution. More precisely, we configured the sampling process with:

Chains	**Tunes**	**Draws**
2	1000	2000

in other words, it means that we run 2 chains in parallel, tune each chain for 1000 iterations and then sample 2000 values from the posterior per chain, giving a total of 4000 posterior samples.

The posterior distributions capture the uncertainty in the estimated parameters and allow us to have a general overview of the most credible values for the estimated identities. Furthermore, by comparing the behavior of the chains we could monitor their convergence to similar posterior distributions, proofing the stability and reliability of the estimates.

By examining the trace obtained through the sampling process, we have observed that all the distributions exhibited an $\hat{R}$=1, implying that the sampling process effectively explored the parameter space. This result enhances the robustness of our conclusions.

The mean of these distributions serves as the point estimate for the intercept and the coefficients, which are then used to construct a linear regression usable for prediction purposes.

3.2.1 Results

The results of the model just introduced on the new, unseen test data are shown in Figure 3.2. We can observe that the model successfully captures the main patterns, as it it able to follow the predictions quite accurately. We have to observe, however, that it becomes less accurate with respect to peaks and troughs, as highlighted by the red-colored areas inside the picture.

The table below provides a more complete overview about the metrics we evaluate our model on. It is crucial to mention that, while in Figure 3.2 the y scale is based on the MinMax scaled values, the table 3.1 on the right expresses the MAE and the RMSE in real terms, i.e. using the target variable's unit of measure. This duality will hold for the subsequent chapters as well.

One potential improvement for this phenomenon is to specify different prior distributions potentially able to account for heteroskedasticity.

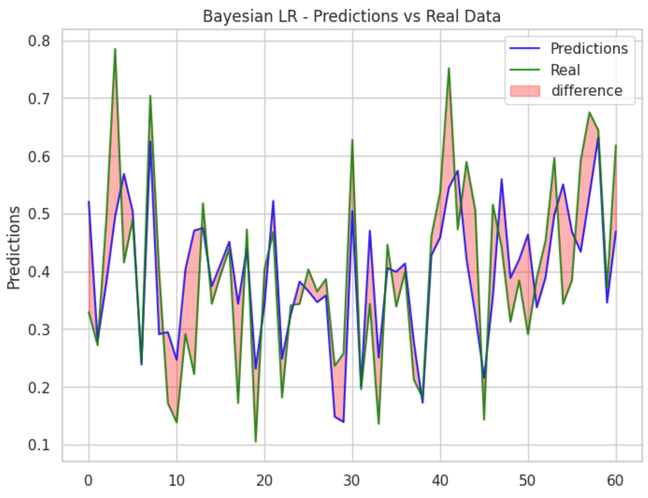


Figure 3.2: Prediction Behavior

Metric	Value
R^2	0.5143
BIC	107.4487
MAE	18.7946
RMSE	23.3005

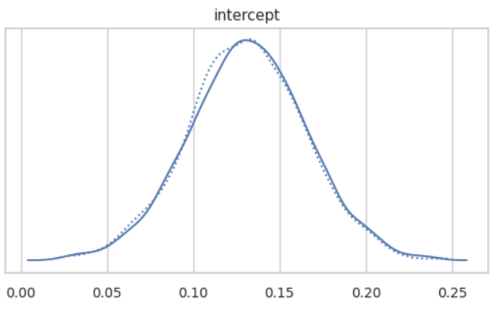
Table 3.1: Metrics summary

3.3 Bayesian Linear Regression - Student-T Priors

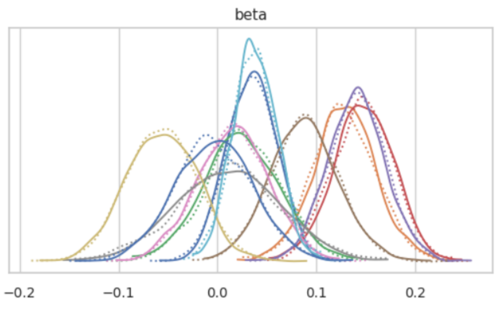
The second version of the Linear Regression family differs from the previous one in terms of the prior distributions specified with respect to the intercept and the variables' coefficients. More precisely, we modeled the priors as Student-T distributions with the scope of allowing for heavier tails compared to the normal distribution, making it more robust to outliers and extreme values in the data.
In a more rigorous setting, the prior distributions are written as follows:

$$Int, \beta_i \sim t(\nu{=}1, \mu{=}0, \sigma{=}10) \quad i = 1, \ldots, 11$$

We therefore determined the estimated regression coefficients following the same approach mentioned in Chapter 3.2. In order to simplify the comparison with the previous Bayesian specification, we maintained the same configuration for the sampling process, in terms of draws, tuning, and number of chains.



(a) Intercept posterior



(b) Coefficients' posteriors

Figure 3.3

Similar to what we encountered with the first model, the two chains outputted converging estimates for the intercept and the coefficients. Also in this case we observed $\hat{R}$=1 for all the distributions, allowing us to derive the same conclusions in terms of the reliability of the estimates.

3.3.1 Results

The following image, together with the metrics summary Table 3.2, depicts the performance of this second version of the Bayesian Linear model. In order to facilitate the comparison with its previous companion, also the predictions of the Normal-priors model is included in the line plot.
It is clearly visible that there is no significant difference between the two proposed Linear Model versions. While we expected to obtain slightly different results from this second experiment, the comparative analysis between the two proposed Linear Model versions yielded no substantive deviation in predictive efficacy.

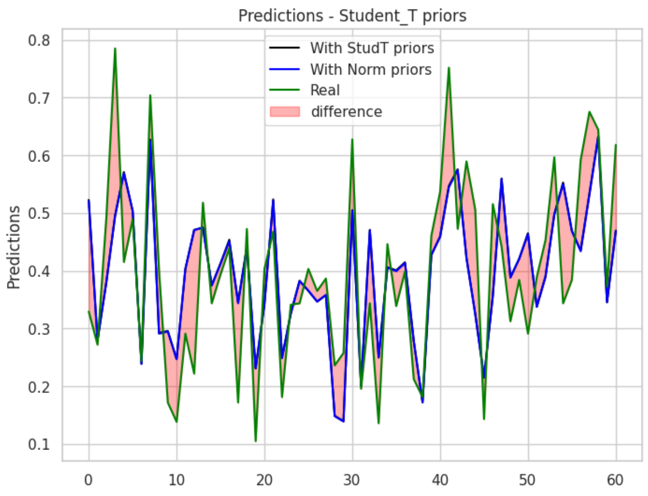


Figure 3.4: Prediction Behavior

Metric	Value
R^2	0.5151
BIC	107.5702
MAE	18.7993
RMSE	23.2986

Table 3.2: Metrics summary

3.4 Hyperparameter Tuning

Concerning the model introduced in Chapter 3.2, we perform hyperparameter tuning on the mean and standard deviation of the normal priors, and the number of MCMC samples and tuning iterations.

The Bayesian optimization algorithm we adopted is based on Gaussian Processes and it iteratively selects hyperparameter configurations, evaluates their performance with respect to the MAE, and updates a probabilistic surrogate model to guide the search toward promising regions of the hyperparameter space.

Despite leveraging Bayesian optimization to fine-tune the hyperparameters of our model, the resulting configuration did not yield a significant improvement in performance compared to the previous models. However, one noteworthy improvement we noticed for the BIC, which is lower than all the previous counterparts. Our interpretation of this outcome is that the initial set of hyperparameters was already close to their optimal settings, leaving little room for improvement.

μ	σ	Draws	Tunes
6.7	2.7	2475	6770

Table 3.3: Hyper tuned values

Metric	Value
R^2	0.5155
BIC	107.2781
MAE	18.7910
RMSE	23.3011

Table 3.4: HPT Metrics summary

4 Ridge Regression

Ridge regression is a variant of linear regression whose aim is to address multicollinearity and overfitting. Ridge regression offers a solution to these problems by introducing a penalty term to the traditional least squares objective function utilized in linear regression.

In the analysis of our linear model, we encountered a minor potential issue related to multicollinearity. To deepen it, we analyzed the Variance Inflation Factor (VIF), which quantifies the severity of multicollinearity by measuring how much the variance of the estimated regression coefficients is inflated due to multicollinearity, and we identified two predictors with VIF values exceeding 5.

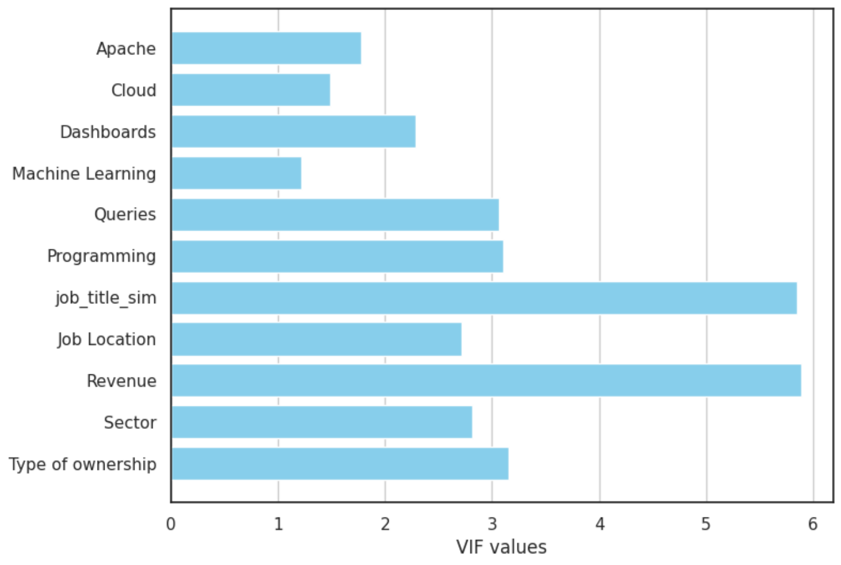


Figure 4.1: Variance Inflation Factor

While the conventional threshold after which a variable becomes troublesome is typically set at a VIF value of 10, some researchers advocate for a more conservative threshold of 5. To address this uncertainty and mitigate the effects of multicollinearity, we opted to employ Bayesian Ridge Regression. As a regularization technique, it introduces a penalty term to the regression coefficients, thereby shrinking them towards zero and reducing the impact of multicollinearity. This penalty term penalizes large coefficient values, effectively shrinking them toward zero.

The Bayesian Ridge model offered by the SciKit-Learn library[1] fits the data using

[1]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.BayesianRidge.html

Gamma priors over the regularization parameters λ and α. While λ controls the amount of regularization to be applied to the coefficients, α controls the uncertainty of the model.

4.1 Results

After implementing ridge regression in our analysis, we observed a slight but noticeable improvement in prediction accuracy, as indicated by the reduction in Mean Absolute Error (MAE) to its lowest value observed thus far.

The decrease in MAE suggests that ridge regression has successfully addressed the minor issues associated with multicollinearity in our dataset Moreover, since its results are on the same order of magnitude with respect to the previous counterparts, it proved that our overall approach is to be considered solid.

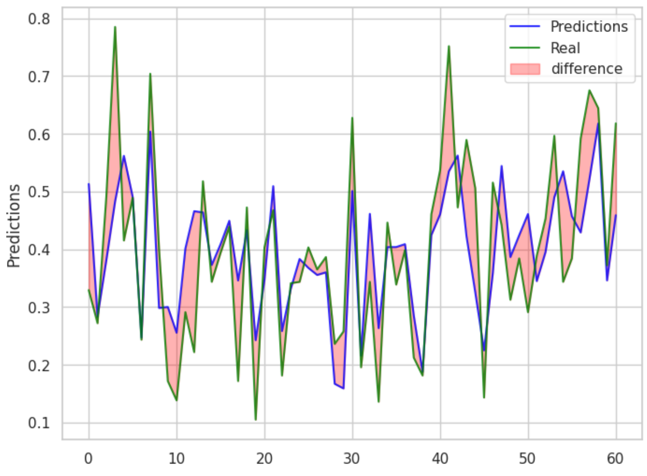


Figure 4.2: Bayesian Ridge Predictions

Metric	**Value**
R^2	0.5126
BIC	107.7497
MAE	18.7683
RMSE	23.36

Table 4.1: Metrics summary

5 Regression Tree

The second family of models we propose involves Regression Trees. We chose it to capture nonlinear relationships and handle complex interactions within the data. We explored the Bayesian Regression Tree by implementing a Perpendicular Regression Tree algorithm, which does not apply Markov Chain Monte Carlo and does not require a pruning step. Our model was built using modules from UBS-IB GitHub Repository, which was inspired by the algorithms published in *A Bayesian Decision Tree Algorithm* by G. Nuti et al (both provided in References).

5.1 Bayesian Regression Tree

For the regression tree model in our project, we had the choice between two different Bayesian approaches: the *Perpendicular Regression Tree*, which uses perpendicular splits of the data space, and the *Hyperplane Regression Tree*, which uses splits with any orientation by optimizing the position of splitting hyperplanes. While the first option is simpler and faster because its splits are constrained to the coordinate axes, the second option can model the data in a more complex way by allowing splits in any direction.

After considering the trade-off between complexity, computation time, and flexibility of modeling, we decided to use the Perpendicular Regression Tree for our experiments since implementing the perpendicular splits would be easier and quicker to test compared to the hyperplane approach, which involves additional optimization steps during training.

5.1.1 Perpendicular Regression Tree

We begin by defining prior distributions for the model parameters. These priors are based on informative statistics calculated from the training data. Specifically, we compute the mean μ and standard deviation σ of the target variable (Salary) from the training set (σ is scaled down by a factor of 10 to ensure a reasonable prior spread).

Additionally, we set the number of prior pseudo-observations (κ) to 1, which reflects our initial uncertainty about the parameters.

Using these prior specifications, we calculate hyperparameters for the prior distributions of the regression coefficients. We compute κ, α, β, and τ, which correspond to the parameters of the Normal- Gamma distribution prior.

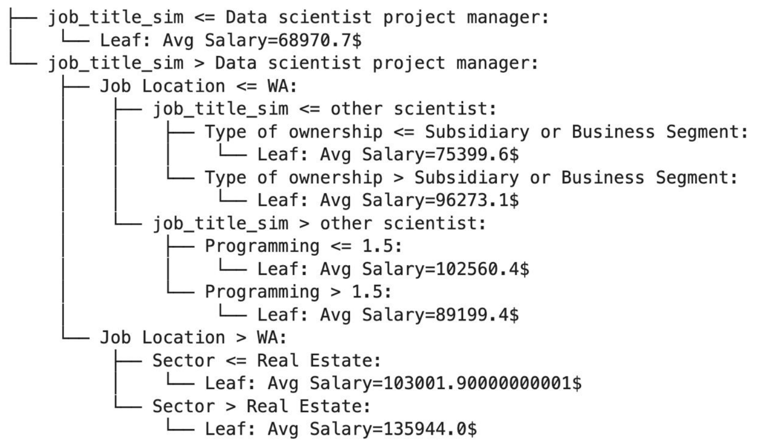
$$\alpha = \frac{\kappa}{2}, \quad \tau = \frac{1}{\sigma^2}, \quad \beta = \frac{\alpha}{\tau} \to \text{Priors} = [\mu, \kappa, \alpha, \beta]$$

Here's how these priors are used:

- μ: It reflects our belief about the average value of the target variable before observing any data.
- κ: It controls the strength of our belief about the mean.
- α: Represents half of the prior pseudo-observation count used to compute the sample variance. It influences the shape of the prior distribution of the regression coefficients.
- β: Represents the ratio of α and the prior pseudo-observation sample variance (τ). It determines the spread of the prior distribution.

With the prior distributions established, we initialize the Bayesian Regression Tree model using the `PerpendicularRegressionTree` class.

We set model parameters such as `partition_prior` $= 0.9$ (prior probability of splitting a node) and `delta`$=0$ (regularization parameter) to fine-tune the model's behavior.



```
├── job_title_sim <= Data scientist project manager:
│   └── Leaf: Avg Salary=68970.7$
└── job_title_sim > Data scientist project manager:
    ├── Job Location <= WA:
    │   ├── job_title_sim <= other scientist:
    │   │   ├── Type of ownership <= Subsidiary or Business Segment:
    │   │   │   └── Leaf: Avg Salary=75399.6$
    │   │   └── Type of ownership > Subsidiary or Business Segment:
    │   │       └── Leaf: Avg Salary=96273.1$
    │   └── job_title_sim > other scientist:
    │       ├── Programming <= 1.5:
    │       │   └── Leaf: Avg Salary=102560.4$
    │       └── Programming > 1.5:
    │           └── Leaf: Avg Salary=89199.4$
    └── Job Location > WA:
        ├── Sector <= Real Estate:
        │   └── Leaf: Avg Salary=103001.90000000001$
        └── Sector > Real Estate:
            └── Leaf: Avg Salary=135944.0$
```

Figure 5.1: Regression Tree Output

Metric	**Value**
R^2	0.1857
BIC	139.0584
MAE	23.1483
RMSE	30.1943

Table 5.1: Metrics

After fitting the model, we evaluate its performance and characteristics. We inspect the tree structure, to understand its interpretability. For instance, at the root, the model splits based on job title. Subsequent splits consider job location, type of ownership, programming proficiency levels and Sector.

Finally, we assess the model's predictive accuracy on the test set using the metrics in Table 5.1. We noticed surprisingly that this model had bad fitting properties, as well as poor predictive power, such as a higher BIC and a lower R^2 compared to the linear models.

Our interpretation of this negative performance is that the Perpendicular approach couldn't catch the data's complexity.

6 Conclusions

As a final discussion of this project, we propose a feature importance analysis on the best-performing model in terms of MAE, i.e. the Bayesian Ridge Regression. This model will also be used in our above-mentioned web-based application.

First of all, we emphasize that almost every skills-related regressor has a positive impact on the *Salary* outcome. This is not surprising, as having more talent is typically translated into higher compensation levels, for the skills provide the employee with the set of knowledge necessary to be an expert in the field he operates in. However, it is noteworthy to underline the estimated negative impact of the variable *Dashboards*: a higher level of expertise in this category is associated with lower salaries, on average.

Due to the Target Encoding mentioned in Chapter 2.2, the coefficients associated with the categorical variables are not easily and directly interpretable. Nonetheless, since their magnitude plays a significant role in predicting the salary, we can conclude that they are highly influential.

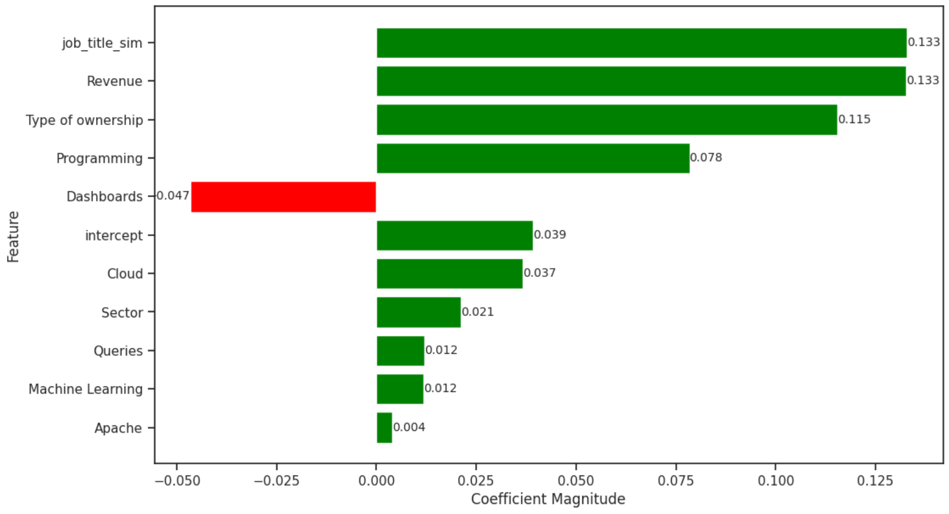


Figure 6.1: Feature Importance based on Coefficients

6.1 General Evaluations

The main insights we derive at the end of our analysis are the following:

- The linear models demonstrated higher predictive precision when compared with the non-linear regression tree, for which we expected more promising results.
- Ridge Regression established itself as the best model in predictive accuracy, even though the difference with the other linear models is negligible.

- Hyperparameter tuning didn't provide a substantial contribution to the improvement of our linear regression model, probably because of the similarity between the hypertuned values and the base ones, other than due to limited computational resources.
- As expected, higher levels of expertise are translated into higher salaries, with a particular accent for *Programming* and *Cloud*.
- The best MAE obtained, namely 18, represents a desirable starting point, yet improvable, as it captures a non-negligible uncertainty over the final predictions.

6.2 Potential Improvements

Multiple areas for development deserve our attention. First, acquiring more computational resources would allow us to investigate a larger range of model configurations and tune our algorithms more precisely. Secondly, augmenting our data with more extensive and complete data would strengthen the reliability of our predictive results, aiming at reducing the uncertainty captured by the MAE. At last, we do believe that including additional features such as gender, age, years of experience, and other job-related skills would contribute positively to the predictive power of our models.

7 References

- Kaggle Dataset
- Our Project's GitHub Repository
- UBS-IB Implementation
- Giuseppe Nuti, Lluís Antoni Jiménez Rugama, and Andreea-Ingrid Cross. A bayesian decision tree algorithm, 2020.