**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Title:** | Data preparation and visualization, Machine Learning, Statistics and Programming |
| **Assessment Title:** | Foreign Nationals in Employment |
| **Lecturer Name:** | David McQuaid, Muhammad Iqbal, Taufique Ahmed, Sam Weiss |
| **Student Full Name:** | Fernando Tupa Leniz |
| **Student Number:** | 2023341 |
| **Assessment Due Date:** | 10/11/2023 |
| **Date of Submission:** | 10/11/2023 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

**Introduction**

In today's data-driven world, the application of machine learning models, statistical knowledge, and my coding skills in Python using Jupyter notebooks has become central to my work in the field of data analytics and machine learning. The objective of this CA aligns perfectly with my professional interests, as it entails predicting the weekly salaries of employees within the industrial sector. To accomplish this task, I'll be using a robust and informative dataset from the Central Statistics Office (CSO) of Ireland, which contains a wealth of information about employees, their characteristics, and their earnings.

You can explore the code and documentation related to this project in my GitHub repository: <https://github.com/FernandoDataAnalitycs> .

The ability to predict weekly salaries within the industrial sector is of great importance to a wide range of stakeholders, including policymakers, businesses, and employees themselves. Accurately forecasting salaries is not just a statistical exercise; it holds the potential to uncover valuable insights into wage determination, wage disparities, and labour market dynamics.

In this first CA, I will delve into data preprocessing and visualization techniques to clean and explore the dataset, making it ready for machine learning model development.

The role of data preparation and visualization is fundamental in uncovering patterns and relationships within the dataset. Visualization tools will enable me to gain a deeper understanding of the data's underlying structures and correlations, thereby informing feature selection and model creation. In addition, I will look for the best way to execute coding using python libraries or methods that can help me to type less and do more.

This CA not only serves as a practical exercise but is closely aligned with the work I do in my professional capacity. By the end of this project, I aim to create a machine learning model capable of predicting weekly salaries with a high degree of accuracy. This will contribute to a deeper understanding of the labour market in Ireland and serve as a powerful tool for informing labour market policies and guiding organizations in making data-driven decisions regarding employee compensation.

Throughout this assignment, I will apply various machine learning algorithms, statistical techniques, and my coding skills to develop and fine-tune our predictive model. The ultimate goal is to leverage data-driven insights to make a tangible impact in the field of labour market analysis.

**Data preparation and visualisation part:**

**1)**

In my Exploratory Data Analysis (EDA) process, I utilized several methods to thoroughly examine my dataset for irregularities.

Initially, the shape() method allowed me to understand the dataset's dimensions, providing insights into the number of rows and columns. This served as a foundational step in comprehending the dataset's size and structure.

The dtypes() method became invaluable as it revealed the data types associated with each column. This information was crucial in understanding the nature of variables, distinguishing between integers, floats, and strings.

The info() method played a key role in presenting a succinct summary of the dataset. It encapsulated essential details such as the number of non-null values and overall memory usage, offering a quick assessment of data completeness and highlighting potential discrepancies in data types.

To delve deeper into data completeness, the count() method proved helpful by indicating the number of non-null values in each column. This directed my attention to columns with lower counts, prompting further investigation into potential issues.

To identify missing values, I employed the isnull() with the method sum(), which provided a count of null values in each column. This was instrumental in pinpointing columns with significant missing data, guiding me in making informed decisions on how to handle these gaps, whether through imputation or removal.

I deliberately applied the drop() method to eliminate certain columns, such as 'Statistic Label' and 'UNIT', from the dataset. The rationale behind this decision was rooted in the understanding that these specific columns held information that was deemed irrelevant for the purposes of my analysis. 'Statistic Label' and 'UNIT' might contain descriptors or units of measurement that, while potentially crucial in other contexts, were deemed unnecessary for the specific insights and patterns I sought to uncover during the exploratory phase.

In addition, I employed the rename() method to enhance the interpretability of column names, ensuring a more intuitive comprehension of the dataset, particularly in the context of deploying machine learning models. By adopting more descriptive and standardized column names, I not only streamlined the EDA process but also set the stage for a more seamless integration of the dataset into machine learning workflows, where well-labeled features contribute to the overall effectiveness and interpretability of the models employed.

By combining these methods, I gained a comprehensive understanding of my dataset, laying the groundwork for subsequent data preprocessing and more advanced analytical exploration. This thorough examination is crucial for identifying and addressing irregularities, ensuring the reliability and quality of the data for further analysis.

I identified and addressed the presence of null values within the dataset. Null or missing values can significantly impact the reliability and accuracy of any analytical endeavor, potentially leading to skewed insights or flawed conclusions. To mitigate this issue, I strategically applied the fillna() method, a fundamental data preprocessing technique, to systematically fill in the missing values with appropriate data points. By doing so, I ensured that the dataset maintained its structural integrity and completeness, laying a robust foundation for subsequent analyses. This meticulous handling of null values not only enhances the overall quality of the dataset but also serves as a crucial step in fostering a more accurate understanding of the underlying patterns and trends during the exploratory phase. Through such proactive measures, the dataset becomes more conducive to meaningful insights and facilitates a smoother transition into subsequent stages of analysis and modelling.

Furthermore, recognizing the necessity for numerical inputs in machine learning models, I employed the Label Encoder method as an integral component of my data preprocessing efforts. Given that many machine learning algorithms require numerical representations for categorical variables, the Label Encoder proved indispensable in transforming string-based categorical data into a format compatible with these models. By systematically assigning numerical labels to distinct categories, I not only overcame the inherent limitation of using string data in machine learning models but also established a standardized and computationally viable representation for categorical features. This crucial step in the preprocessing pipeline ensures that the dataset is well-suited for a wide array of machine learning algorithms, enhancing its adaptability and paving the way for more accurate and effective model training during subsequent stages of analysis and exploration.

**Machine Learning part:**

**1)**

In this first part, I think CRISP-DM (Cross-Industry Standard Process for Data Mining) is better than KDD (Knowledge Discovery in Databases) or SEMMMA (Sample, Explore, Modify, Model, and Assess) for a data science project. I will justify with some explanation and examples:

First of all, CRISP-DM is considered an industry standard and is widely adopted in the field of data science. Based on its six phases like business understanding, data understanding, data preparation, modelling, evaluation and deployment. It provides a structured approach that is recognized and accepted across industries. In contrast, KDD lacks the practical guidelines and widespread acceptance that CRISP-DM offers. SEMMA, developed by SAS for its software products, is less known outside of SAS-centric environments, limiting its applicability in organizations using various tools. For example, consider that a huge corporation wants to adopt a standardized data science approach across various departments. In such a case, CRISP-DM's industry-wide recognition and acceptance make it the preferred choice.

Secondly, CRISP-DM is a flexible methodology that can be adapted to different types of data science projects and is not tied to specific tools or techniques. KDD, is more theoretical and less prescriptive in terms of specific project phases and techniques. SEMMA is being closely integrated with SAS tools, may not be easily adaptable for organizations using alternative data science platforms. For instance, a startup with limited resources that seeks agility in adopting different data science techniques based on project requirements. CRISP-DM's flexibility allows them to do so seamlessly.

Thirdly, CRISP-DM places strong emphasis on an iterative process, encouraging revisiting and refining earlier stages as new insights and data become available. KDD does not explicitly emphasize iteration and lacks detailed guidance on revisiting stages. SEMMA also lacks a strong focus on iteration, which may be less suitable for projects requiring continuous improvement. For example, in the healthcare industry, where new patient data continually becomes available, a data science project aimed at improving patient outcomes benefits from CRISP-DM's iterative approach.

Finally, CRISP-DM places a strong emphasis on understanding the business problem and aligning data science goals with business objectives. This is crucial for the success of a data science project. KDD, on the other hand, focuses more on data mining techniques and may not provide as strong a link to business understanding. SEMMA does include elements of business understanding but is not as comprehensive as CRISP-DM in this regard. For instance, In the retail industry, where understanding customer behaviour is essential for improving sales and marketing strategies, CRISP-DM's focus on business understanding proves advantageous.

To continue with this first part, I would choose the supervised machine learning as technique for my dataset. It is important to remark the differences between supervised, unsupervised, and semi-supervised learning as I list below:

Supervised Learning:

- Trained on labeled data.

- Learns a mapping from input to output.

- Used for classification and regression tasks.

Unsupervised Learning:

- Works with unlabeled data.

- Identifies patterns or clusters in the data.

- Used for clustering and dimensionality reduction.

Semi-Supervised Learning:

- Uses both labeled and unlabeled data.

- Leverages labeled data to guide learning on unlabeled data.

- Balances supervised and unsupervised learning principles.

According to the differences, the dataset match well with supervised machine learning. The labeled data like *“Type of Employee”,* when there are different kind of employees or “Industry Sector”, which shows different industries. Those columns are considered as categorical data. It is remarkable the purpose of using supervised machine learning when is considered useful for regression tasks in the current dataset.

**2)**

In this second part, I chose Random Forest Regression and XGBoost Regression, because they are both powerful machine learning models with unique advantages when it comes to estimating weekly salaries of industry employees. These models are particularly useful in regression tasks where the goal is to predict continuous numeric values, such as salary amounts.

The first one that I used was Random Forest Regression, because it emerges as a highly effective model for the prediction of weekly salaries in industrial contexts, presenting a multitude of advantages that contribute to its robust performance. Its foundational strength lies in the amalgamation of predictions from numerous decision trees, each constructed on a random subset of the data, leading to a final prediction derived from the averaging or voting mechanism of individual tree outputs. This ensemble approach enhances predictive accuracy, offering a more resilient and precise model compared to a singular decision tree. Furthermore, Random Forest Regression adeptly handles non-linear relationships inherent in the salary determinants, accommodating the complexity of interactions between various factors influencing compensation. Its ability to naturally manage categorical variables, such as type of employee or industry types, without necessitating intricate pre-processing simplifies the modeling process and diminishes the risk of information loss.

Moreover, the model demonstrates a robustness to overfitting, particularly when the ensemble encompasses a substantial number of trees. This resilience proves crucial when dealing with real-world data, notorious for its noise and outliers. Beyond predictive capabilities, Random Forests provide valuable insights into variable importance, elucidating which factors wield the most significant impact on employee salaries within the industrial landscape. This knowledge empowers decision-makers to focus on the most influential determinants when considering compensation structures. Additionally, the ensemble nature of Random Forests lends itself to effective outlier handling, a pivotal aspect in scenarios where anomalies can unduly influence model performance.

The model's user-friendly nature further enhances its appeal, as it necessitates minimal hyperparameter tuning and can be seamlessly integrated into existing workflows. In this case I used RandomizedSearchCV, It is particularly useful when there are a large number of hyperparameters to explore. This hyperparameter tuning method is distinguished by its computational efficiency, as it samples a predetermined number of hyperparameter combinations from a specified search space. It proves to be more computationally efficient compared to GridSearchCV, which exhaustively explores all possible combinations. Another important characteristic is the parallelization, because of the random nature of the search makes it easier to parallelize. Different combinations of hyperparameters can be evaluated independently, leading to faster hyperparameter tuning compared to a sequential approach.

When I started training the model, I got 0.93 of performance. The parameters that I used are:

(n\_estimators = 500, max\_features = 'sqrt', max\_depth = 8, random\_state = 10, min\_samples\_split= 4, min\_samples\_leaf= 2, bootstrap= False).

On the scond time, I just use the class created and I got 0.96 of performance, using:

(n\_estimators=300, max\_depth=7, random\_state=10). In this case, the class RandomForestRegression() used predetermined values if I did not put everything

After applying RandomizedSearchCV, it found the best parameters:

('random\_state': 10, 'n\_estimators': 350, 'min\_samples\_split': 2, 'min\_samples\_leaf': 2, 'max\_features': None, 'max\_depth': 8, 'bootstrap': True), and it got a performance of 0.97. That means that it could be possible to improve the model by using hyperparameters’ tunning.

On the other hand, using XGBoost Regression in the regression task of predicting weekly salaries for industry employees offers several key advantages. XGBoost's ensemble approach contributes to high predictive accuracy by combining the strengths of multiple weak models, making it particularly effective in capturing non-linear relationships and complex patterns in salary data. The algorithm's feature importance analysis provides valuable insights into the factors influencing salary predictions, enhancing model interpretability. XGBoost's robustness to overfitting, efficient handling of missing data, support for parallel and distributed computing, and a plethora of tunable hyperparameters allow for the optimization of model performance, especially in scenarios involving large datasets. Additionally, XGBoost's widespread adoption and community support ensure continuous development and access to valuable resources, making it a powerful and versatile choice for accurate and meaningful salary predictions in real-world applications.

I used GridSearchCV to look for the best hyperparameters. Employing `GridSearchCV` in the context of an XGBoost Regression model offers numerous advantages. Through systematic exploration of a predefined hyperparameter grid, GridSearchCV automates the tuning process, identifying optimal hyperparameter combinations that maximize model performance. This exhaustive search, coupled with cross-validation, ensures robust generalization to unseen data. The approach is computationally efficient, striking a balance between thorough exploration and practical resource use. By optimizing performance metrics tailored to the salary prediction task, such as mean squared error or R-squared, the method enhances the model's accuracy and relevance. The transparency and reproducibility of GridSearchCV provide a documented and communicable process, and fine-tuning hyperparameters allows the XGBoost Regression model to address specific intricacies in predicting salaries, including non-linear relationships and industry-specific nuances. Overall, the systematic and automated nature of GridSearchCV significantly contributes to the development of an accurate, robust, and industry-tailored salary prediction model.

At the beginning, I got 0.988 of performance when I used the hyperparameters :

(n\_estimators = 200, learning\_rate=0.35, max\_depth = 3, min\_child\_weight = 3, subsample = 0.55, colsample\_bytree = 0.7, reg\_alpha=0.5, reg\_lambda=1).

After using hyperparamters I got 0.991. As a result, the best parameters were:

('colsample\_bytree': 0.7, 'learning\_rate': 0.06, 'max\_depth': 5, 'min\_child\_weight': 2, 'n\_estimators': 500, 'subsample': 0.5). Getting a little improvement, the model could use better parameters to get a good performance.

**3)**

In this part, I am going to review and critically examine the machine learning models' performance based on the selected metric and plots.

Before to show the results, it is important to mention the importance of splitting data into separate sets for training, fitting, and predicting is integral to effective machine learning analysis. This practice serves as a safeguard against overfitting, allowing the model to generalize well to new data by evaluating its performance on an independent test set. It facilitates the crucial process of hyperparameter tuning, ensuring that adjustments are made based on the model's performance on unseen data. The separation of training and test sets also prevents data leakage and enhances the model's interpretability, contributing to the development of trustworthy models. Furthermore, it supports advanced techniques like cross-validation, providing a more robust evaluation of the model's generalization capabilities, particularly in cases of imbalanced datasets. In essence, this approach is foundational for building reliable, accurate, and interpretable machine learning models.

The first model that I used was Random Forest Regression, and after train, fit and predict into the model I got this graph:

A graph of a graph with red dots

Description automatically generated with medium confidence

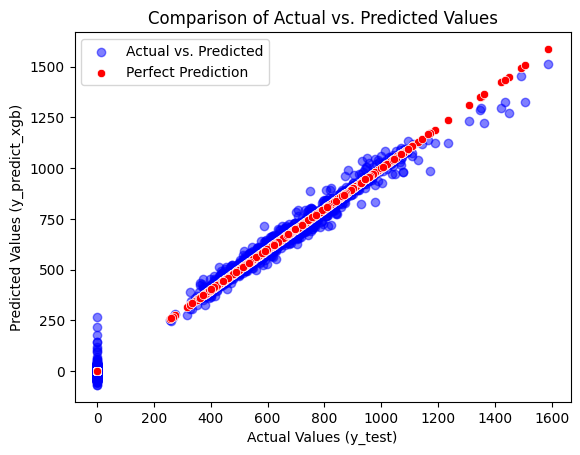
There are “actual vs predicted” values in color blue. That is in the context of a Random Forest Regressor analysis. It shows visually assess how well the model is performing. Based on the perfect prediction line in color red, where most of the blue points are close to the red points. This line represents perfect predictions, where the predicted values exactly match the red points. If it is closer to the red points, the better the model's predictions are. If points deviate far from the line, it indicates that the model's predictions are not accurate.

Another interesting analysis is the distributions of points, If the points form a tight cluster around the line, it is suggesting good model performance. If there was a scattered pattern, it may indicate areas where the model struggles. In case to see outliers could represent instances where the model's predictions are significantly different from the actual values. In this case there no notorious points outside the dispersion show in color blue.

It is important to consider using quantitative metrics to assess model performance, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or R-squared. These metrics can provide a numerical summary of how well the model is performing. Particularly, R-squared (R2) score is a metric that measures the proportion of the variance in the dependent variable (target) that is explained by the independent variables (features). In the context of a Random Forest Regressor, the R2 score can provide insights into how well your model is capturing the variance in the target variable.

The R2 score ranges from 0 to 1. A score of 1 indicates that the model perfectly predicts the target variable, while a score of 0 suggests that the model does not explain any variance. Negative values are possible and indicate that the model is worse than a simple mean-based model. In the moment that I got the R2 using metrics from sklearn, the first one was 0.93 and the second one 0.96. Which means that they have a good performance, because both are close to 1.

The second model that I used was XGBoost Regressor, and after train, fit and predict into the model I got this graph:



Despite of the analysis about dispersion of red points and blue points are almost similar like the previous graph, it is not the same. In the center of the graph there are many blue points close to red points (perfect prediction). In addition, there are more dispersion at the beginning of the red line and at the end as well. Points below the red line indicate cases where the model predicted values that are lower than the actual values. This suggests that the model underestimated the target variable for these instances. Points above the red line indicate cases where the model predicted values that are higher than the actual values. This suggests that the model overestimated the target variable for these instances. It is possible that points significantly away from the red line, in either direction, may be considered outliers. Outliers can represent instances where the model struggled to make accurate predictions.

Even though I talked about R-squared as metric to analyze the performance, I would mention another metric. Mean Squared Error (MSE) is a commonly used metric in machine learning to evaluate the performance of a regression model. It quantifies the average squared difference between the predicted values and the actual values. MSE is always a non-negative value. It reaches its minimum value of 0 when the model perfectly predicts the actual values. As the predictions deviate from the actual values, the MSE increases. In this case I got 1143.86, which means that provides a quantitative measure of prediction error and it is not low. It is better if it is close to 0, because it would show a better predictive accuracy.

On other hand, MSE is often more informative when compared to the MSE of other models or a baseline model. For example, if the previous model with an MSE of 6277.98 at the first try or 3818.92 on the second time, this MSE of 1143.86 might be considered an improvement.

**4)**

In this part, I will demonstrate differences between my machine learning modelling results by using the metric MSE and cross-validation:

A comparison of a graph

Description automatically generated with medium confidence

Cross-validation is a technique used in machine learning to assess the performance of a model and to compare different models. At the moment to compare two machine learning models, cross-validation helps to estimate how well each model is likely to perform on an independent dataset.

In cross-validation analysis using Mean Squared Error, the primary interpretation is that a lower MSE is generally better. The MSE in Random Forest Regression model is 3818.92.However, XGBoost regressor got a better score with 398.44. If we compare different models or algorithms, consider the one with the lower cross-validated MSE as potentially having better generalization performance. It suggests that the model is making more accurate predictions on average across different subsets of the data. For instance, the cross-validated MSE in Random Forest Regression is lower than the second model with 10155.43m against 18111.13.

**Statistics part:**

**1)**

In this part, I am going summarise my dataset clearly, using relevant descriptive statistics and plots.

First of all I used the describe(),because it is useful as a robust instrument for conveying pertinent descriptive statistics essential for comprehensive data understanding. This method furnishes a succinct yet comprehensive summary of key statistical measures for each numerical column within the dataset. The inclusion of metrics such as mean, standard deviation, minimum, maximum, and quartiles facilitates a rapid assimilation of the central tendencies and distributions characterizing the numerical features.

The range ,that it is possible to see on the table ,is a simple but informative measure of the spread of a dataset. A larger range indicates a wider spread or dispersion of values, like “value” column, while a smaller range suggests that the values are more concentrate, like “industry\_sector” column.

The mean and median are both measures of central tendency, but they are calculated differently and can provide different insights into the distribution of data. When the mean and median are close or equal, it suggested that the dataset is symmetrically distributed and not heavily influenced by outliers,for example “industry\_sector”,” type\_of\_employee” and “quarter”. However, if there is a significant difference between the mean and median, it could indicate the presence of outliers or skewness in the data.

Boxplots, histograms, and violin plots are all valuable tools in visualizing and understanding the distribution of data in machine learning analysis. According to the first one:

A graph with a line and a line in the middle

Description automatically generated with medium confidence

It is excellent for identifying outliers in a dataset, which are data points significantly different from the rest, can have a substantial impact on machine learning models. Boxplots it was useful to compare the distributions of different groups or categories within a dataset, providing a quick visual summary of their central tendency and spread.

A graph of a quarter

Description automatically generated with medium confidence

Histograms provide a visual representation of the distribution. They are especially useful for understanding the shape of the data, including whether it is symmetric, skewed, bimodal, etc. In this case, it seems like normal distribution according to its shape.

A blue and white graph

Description automatically generated

Violin plots combine the information from boxplots and kernel density plots. It provide a summary of the distribution's shape, including information about modes and outliers. In the graph I am using the column “type\_of\_employee”.

2)

In this part, I will use the Poisson distribution in order to identify some information about my dataset.

A graph of a number of objects

Description automatically generated

The plot of Cumulative Poisson Probabilities is a visual representation of the cumulative distribution function (CDF) for a Poisson distribution. I used It in the column “quarter”, because Poisson distribution is a discrete probability distribution that describes the number of events that occur within a fixed interval of time or space. The CDF essentially accumulates the probabilities for all possible values up to a specific point x. If the plot rises steeply at a certain point, it indicates a higher likelihood of observing a value close to that point. It started well from 0, but after 20 in x-axis kept in 1.It does not seem to go down.

**3)**

In this part, I will use Normal distribution to identify some information about my dataset.

A graph of a graph of values

Description automatically generated with medium confidence

I applied normal distribution in the column “value”, because Normal distributions are well-characterized mathematically, making statistical analysis and interpretation more straightforward. Parameters like mean and standard deviation have clear interpretations in the context of a normal distribution.

In this case, I used The Yeo-Johnson transformation is a method for power transformation of data, often used to stabilize the variance and make the distribution more symmetric. The Yeo-Johnson transformation is capable of handling both positive and negative values, unlike the Box-Cox transformation, which I could not use it. This makes it more flexible and applicable to a wider range of datasets.

**4)**

In this part, I will explain that the column that I used for Poisson can be transformed to normal distribution. it is possible to transform data from a Poisson distribution to a normal distribution, and this can be achieved through various statistical methods. It can be possible with boxcox or Yeo-Johnson transformation method.

**Programming part:**

Imperative programming:

The for loop I used to show boxplots at the same time. It is associated with imperative programming, which is a programming paradigm that describes computation in terms of statements that change a program's state. Imperative programming uses statements that explicitly instruct the computer on how to perform a task. This kind of loop is a control flow statement in imperative languages used for iterating over a sequence (e.g., an array, list, or range of numbers) and executing a block of code for each iteration. It helped me when I did not want to repeat code and iterate an array of values that match with my dataset. In addition, imperative programming contrasts with declarative programming paradigms, such as functional programming, where the emphasis is on describing what computations should be performed rather than how to perform them. In functional programming, you often see constructs like map, filter, and reduce instead of traditional for loops.

Procedural programming:

When I created the function draw\_histogram(), I thought how it can help me to reuse code. I think the paradigm's emphasis on modularity, achieved through the division of programs into procedures or functions, facilitates effective code management and maintenance, allowing changes to be localized without impacting the entire program. This modular structure also promotes reusability, as procedures can be utilized across different parts of a program or in other programs, contributing to time and effort savings. Additionally, the paradigm's efficiency, attributed to the straightforward, linear nature of the code, is evident in terms of execution speed and memory usage in specific situations. Debugging procedural code is often more straightforward due to the clear flow of execution, aiding in issue identification and resolution. The paradigm's widespread adoption is underscored by the extensive use of procedural languages, contributing to a well-established ecosystem of resources, libraries, and tools that support its application in diverse programming contexts.

Object-Oriented Programming (OOP):

In other context I wanted to use OOP with the class RandomForestRegression(), and make my own methods when I want to use them. I think it is important this paradigm, because it offers a modular and reusable approach to software development, promoting code organization through encapsulation of related functionalities into classes and objects. This modularity enhances readability and maintenance, while the reusability of objects across different parts of a program fosters efficiency. OOP's flexibility and extensibility allow for the addition of new classes without modifying existing code, supporting scalability in evolving projects. With abstraction, OOP simplifies development by focusing on relevant details, and encapsulation ensures data integrity and security. Inheritance facilitates code reuse, and polymorphism enables flexible coding. Easier maintenance is another advantage. It is achieved by isolating changes to specific parts of the code, and OOP's real-world modelling enhances conceptualization. Lastly, the collaborative nature of OOP, with its support for classes and objects, facilitates teamwork, making it well-suited for diverse and complex software development projects.