**CCT College Dublin**

**Assessment Cover Page**

|  |  |
| --- | --- |
| **Module Title:** | MSc. In Data Analytics |
| **Assessment Title:** | * Programming for DA * Statistics for Data Analytics * Machine Learning for Data Analysis * Data Preparation & Visualisation |
| **Lecturer Name:** | * Sam Weiss * Taufique Ahmed * Muhammad Iqbal * David McQuaid |
| **Student Full Name:** | Maria Anany Casas Manosalva |
| **Student Number:** | 2023326 |
| **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. |

|  |
| --- |
| The present information of this project has been collected by The Central Statistics Office in Ireland, from the page <https://data.cso.ie> following the route: “Browse”, “Environment”, “Forestry”, “Afforestation Area”.  The main objective of this project is to analyse historical rates of afforestation in Ireland and accurately predict the area that will need to be afforested from 2023 to 2030.  Ireland is one of the least forested countries in the European Union, with current levels of 11.6% (808,848 hectares) of area covered by trees compared to the EU average of 38.3% covered by forest. There are several reasons for this, but the main one is the historical indifference towards forestry. When Ireland became independent from Great Britain in 1922, only 1.4% of the country was covered in forest and in the following decades very little effort was made to change this situation. It was not until the 1970s that a concerted afforestation campaign began. Over the last 40 years, the total area of land under forestry has increased to the aforementioned 11.6% that Ireland currently has. Unfortunately, this amount is nowhere near the amount needed for Ireland to meet its climate change targets. To meet its climate action commitments, Ireland will need to plant more than 8,000 hectares of forest each year until 2050. This will raise the country's land afforestation rate to more than 18% (although, according to the Environmental Protection Agency, this will need to be closer to 24% today - https://www.euroforestireland.ie/news-posts/forestry-planting-rates-must-exceed-8000ha-per-year-to-meet-climate-targets-epa). The problem with Ireland trying to achieve these targets is that current levels of forestry in Ireland are at the lowest they have been in 20 years. CSO data shows that the 8 000 hectares target, which is included in the Government's Climate Action Plan, is being missed by a significant margin. According to figures from the CSO's first forestry statistical report, Afforestation Area 2021, the rate of tree planting fell from 6,947 hectares in 2007 to just 2,016 hectares in 2021. Only 1,400 have been planted so far this year. hectares of forests. Significantly, the share of forest plantations by farmers in 2021 was only 18 percent, down from 97 percent in 2014, according to the report. County Cork had the largest forest area each year from 2007 to 2016 and from 2019 to 2021. Cork accounted for 17% of the total forest area in 2021, followed by Roscommon (9.4%), Clare (8.6%) and Cavan (7.9%). Despite government targets, the amount spent on forestry last year (€74 million) was less than that spent in 2002. To counter this worrying trend, the government has launched a new €1.3 billion forestry program that the European Commission has recently approved. |

|  |
| --- |
| The following report will describe each line of code performed in the Python document explaining the meaning of each of them; for example In[1] is the line of code 1, In[10] is the line of code 10, found in the Python file.  Programming for DA  First of all, it is necessary to do Exploratory Data Analysis (EDA) which is a methodology used in data science and statistical analysis to explore, analyse and visualise datasets to understand their structure, patterns and main characteristics. To do so, we will follow next steps: loading the data, exploring the information, debugging the data, final data processing, analysing the data.   1. Loading the data   In [1]: Import Pandas library which is used for data manipulation and analysis. It works with DataFrames which allow you to efficiently load, clean, transform, and analyse data.  In [2]: Read the CSV file named 'Afforestation\_area.csv' and store the data in a DataFrame named 'afforestation\_df' (convert the CSV file in a DataFrame).   1. Exploring the information:   In [3]: Display the object type of the 'afforestation\_df' which should be a DataFrame.  In [4]: The shape method shows the number of rows and columns of the DataFrame (5184 rows, 7 columns).  In [5]: The info method describes the data type of each column and the number of non-null values of each one. We can see that there are 5184 entries and 3687 non-null in the ‘VALUE’ column, so we have to decide what we will do with this missing values.  In [6]: the head method shows, in this case, the first 10 rows of the DataFrame.   1. Debugging the data:   In [7]: Delete the rows that contain 'Total Afforestation' since I want to analyse the plantations carried out for each specie and forest owner instead of analysing them in their entirety, this is to avoid generating redundancy or noise.  Filter the DataFrame to remove rows where the columns 'Species' or 'Forest Owner' contain 'Total Afforestation' and update the DataFrame with the requested conditions.  In [8]: Remove columns 'Statistic Label' and 'UNIT' due to they both have the same value: 'Afforestation Area' and 'Hectares' respectively.  The drop method deletes rows or columns in a DataFrame, axis=1 (remove columns instead of rows), inplace=True (modify the DataFrame in place).  In [9]: Rename the columns 'VALUE' and 'Forest Owner' creating a dictionary for then use the method rename to rebrand columns named 'VALUE' to 'Value\_ha' and 'Forest Owner' to 'Forest\_Owner'.   1. Final data processing   It can be shown that there are many NaN or null values in the 'Values\_ha' column (in the csv file they appear as empty data) and most of this information comes from 'Non-Farmer' and 'Public Sector'. Initially, I decided to replace these values to 0 because the afforestation mainly comes from 'Farmer' who receive loans for this work, but when I did the normalization, I got a high frequency of 0 values (about 1000 values) so it is better to remove the rows of null values to avoid this interference.  In [10]: The dropna method drops rows or columns with NaN values in a DataFrame, axis=0 (remove rows instead of columns).  In [11]: Create a variable cero\_count to determine how many zero values there are in the 'Value\_ha' column.  In [12]: Create a dictionary to change some names shown in the 'County' and 'Forest\_Owner' columns.  In [13]: The replace method substitutes some values in the ‘County' and ‘Forest\_Owner' columns according to those found in the dictionary created.  In [14]: Call the info method again to see the new number of rows obtained and if there are still null values, observing that the DataFrame contains 1594 entries or rows but the index values start from 5 to 5182. It means that the indexes need to be reset since rows have been deleted.  It can also be seen that the data type of each column is according to its characteristics and we don't need to perform a conversion.  In [15]: The method reset reboots indexes to start from 0, drop=True (clear all indexes).  Verbose=False (it only will display range index, since it displays specific information)   1. Analysing the data   In [16]: Import Numpy library which is used for used for mathematical calculations, statistics and lineal algebra operations. It works with n-dimensional arrays that are suitable for numerical operations.  Import Matplotlib library that is a powerfull tool used for creating 2D visualizations, plots and for representing data visually.  Import Seaborn that is a data visualization library built on top of Matplotlib and it simplifies the creation of informative and visually appealing statistical graphics.  In [17]: Create a Bar Chart to visualize the “Total Afforestation per County from 2007 to 2022”, except for Ireland (it represents the sum of all counties and we do not want to generate redundancy), by following these steps:   * Make a graph and set its size to 12 (wide) x 10 (high) * ‘Filtered1’ variable is a boolean string that contains True if (afforestation\_df['County'] != 'Ireland') * Filter the data in the 'County' column except 'Ireland' and use the groupby method to calculate the sum of the values in the 'Value\_ha' column for each 'County' group * Generate a Bar Chart with the data contained in the 'afforestation\_county' variable, kind='bar' (create a Bar Chart), figsize=(12, 6) (Create Bar Chart dimensions: 12 (wide) x 6 (high)) * plt.title (set the graph title as "Total Afforestation by County from 2007 to 2022" with a font size of 20) * plt.xlabel (set the x-axis label to "County" with a font size of 15) * plt.ylabel (set the y-axis label to "Afforestation (ha)" with a font size of 15) * plt.xticks(rotation=90) (rotate the x-axes labels by 90 degrees to improve readability) * Prop = {'size': 10} (set the font size to 10) * Display the Bar Chart   In [18]: Create a Bar Chart to visualize the “Total Afforestation by Counties per Year”, except for Ireland, by following the same structure as the previous Bar Chart with the difference that the data is grouped by ‘Year’ instead of ‘County’.  In [19]: Generate a Histogram with Matplotlib library to view the distribution of values. The steps are shown below:   * Make a graph and set its size to 12 (wide) x 10 (high) * Create a Histogram with the name 'Value\_ha' on the x-axis, bins=20 (number of bars or bins in the graph), color='green' (set the color of the bars to green), edgecolor='black' (set the color of edges to black), linewidth=1 (set the linewidth of edges to 1)   In [20]: Create a Histogram with Seaborn library to display the distribution of values, by following a similar structure as the previous Histogram.  In [21]: We can observe that the data is completely right skewed (Asymmetric Distribution) and in order to see the outliers vert clearly, we can use Boxplot with Matplotlib library to visualize possible atypical values.   * Make a graph and set its size to 12 (wide) x 10 (high) * Create a Box Plot with the name 'Value\_ha' on the x-axis, vert=False (create a horizontal boxplot because by default boxplots are vertical)   In [22]: Use a-score method to remove outliers. First, we need to identify the upper and lower limits by adding or subtracting the mean with three times the standard deviation.  In [23]: Find the outliers that are outside the range of the upper and lower limits; in other words, values greater than the upper limit and less than the lower limit.  In [24]: Identify the number of outliers by subtracting the length of the original DataFrame from the length that is within the limits found.  There are not many outliers (24) so, in this case I won't delete them because I would be removing information described in Ireland from 2007 to 2022 (Afforestation in hectares) and I need this information to train the machine and create a proper model. |

|  |
| --- |
| Statistics for Data Analytics  In [25]: Binomial Distribution: This distribution models success or failure events and is used for discrete data.  If I apply Binomial Distribution my probability of success would be 'planting trees' and failure 'not planting trees', in a specific area measured in hectares, which would be an unrealistic case, but let's see what the distribution looks like.  Binomial Distribution Graph  It can be observed that the Real Distribution differs from the Binomial Distribution, so it is not convenient to use it.  In [26]: Poisson Distribution: This distribution is used to model events that occurred in a time interval.  In the case of my data, I could model the tree planting rate per year.  Poisson Distribution Graph  It can be seen that both distributions are similar and follow the same trend, although the Poisson Distribution presents larger peaks than the Real Distribution.  Later we will make use of the Normal Distribution to decide which of them will be suitable for generating prediction models.  In [27]: The correlation model is a statistical measure that describes the relationship between two or more sets of data.  The corr method displays what kind of relationship exists between the numerical values of the DataFrame columns, in this case between the 'Year' and 'Value\_ha' columns  In [35]: Normal Distribution: The normal distribution, also known as the Gaussian distribution, is a continuous probability distribution that is symmetric about its mean, with most observations clustering around the central peak and the probabilities of values farther from the mean decreasing equally in both directions.  Generate a Histogram with the transformed numerical values using Seaborn library on a smaller scale (from 0 to 1000 hectares).  Now we can take these values to a Linear Regression Model.  In regards to the Binomial and Poisson Distributions, the Normal Distribution, with the transformed values, fits my data better. |

|  |
| --- |
| Machine Learning for Data Analysis  Machine Learning (ML) is a branch of artificial intelligence that allows the progressive improvement of tasks through machine learning with computers, which through algorithms and models can learn patterns and make decisions without being explicitly programmed. ML can be classified into 3 categories:   1. Supervised learning: the algorithms work with labeled data trying to find a function that, given the input variables, assigns them the appropriate output label.   This type of learning is divided into Classification (K-nearest neighbors, Naive Bayes, Support Vector Machines (SVM), Neural Networks, Random Forest) and Regression (Linear Regression, Non-linear Regression, Random Forest, Decision Tree).   1. Unsupervised learning: occurs when labeled data is not available for training. We only know the input data, but there are no output data that correspond to a certain input.   This is divided into Clustering and the models used are: K-means, Gaussian Mixture, Hidden Markow Model, Neural Networks.   1. Reinforcement learning: This type of learning is based on improving the response of the model using a feedback process that it obtains from the outside world in response to its actions.   For my project, I want to predict the total afforestation in Ireland per year (Value\_ha will be the variable “y” to predict).  The type of Machine Learning I will use for this task is Supervised Learning since I have labeled information where I can train the machine to predict future values of afforestation in hectares. I will also use the Regression Model since the results I want to obtain are numerical (hectares afforested = ha\_value).  In [28]: Warnings is a library used to suppress warning messages during code execution.  In [29]: We can also visualize the correlation using the Seaborn library to generate Heatmaps in colored shapes, which shows the correlation between the numeric variables of the DataFrame, annot=True (show the values within each cell of the Heatmap cell), fmt='.2f' (display the values in cells with two decimal places), linewidth=2 (set the linewidth of edges to 2).  Although the correlation is not high, it is important to take into account other techniques to look for relationships between variables, since the correlation measures "linear relationships" but there are machine learning models such as Decision Trees or Polynomial Regression models that can capture "non-linear relationships".  In [30]: Display the Histogram on a smaller scale (from 0 to 1000 hectares) using the DataFrame by following the next steps:   * Create a histogram using the DataFrame, where 'x' is used on the x-axis of the histogram, 'kde=True' (argument adds a Kernel Density Estimation (KDE) to the plot, KDE is a smothed way of representating the distribution of the variable) * Set\_size\_inches is a method used to adjust the size of the figure to be 20 (wide) x 10 (tall) inches * Set\_xlim is a method to limit the x-axis to a maximum of 1000 * Display the Histogram   The graph shows a Non-Uniform Distribution (Asymmetric Distribution) of Value\_ha ('Afforestation in hectares') and for Linear Regression Models we need Symmetric Distributions.  For obtain this, we need to convert this Asymmetric Distribution to Symmetric Distribution (Gaussian Bell or Normal Distribution).  To Normalize the data we also need to verify that the distance between minimun and maximun frecuency value is significant (scattered data) that's where we can apply the Normal Distribution.  In [31]: The describe method shows descriptive statistics of a DataFrame such as mean, standard deviation, percentiles, maximum and minimum values, among others.  The table displays a large difference between min (0) and max (4928) of Value\_ha, so in this case I'll proceed to adjust my data to a Normal Distribution.  In [32]: FunctionTransformer transforms real values to transformed values using logarithmic function and to return to the initial values, we transform the logarithmic function into an exponential function (inversely proportional).  Np.log1p (creates a transformer object that applies the 'log1p' function from the NumPy library, 'validate=True' (argument ensures that the transformation is validated).  The transform method transforms numeric columns, excluding object data types, into logarithmic function.  In [33]: The method select\_dtypes(exclude=['object']) selects the columns from the DataFrame that don't have an object data type.  The method concat joins two DataFrames along the columns or rows (in this case columns). This line combines the categorical columns with the transformed numerical columns.    In [34]: After the transformation we create a Heatmap to see the new correlation. It can be seen a slight increase in the value of the correlation, but it is not significant.    In [35]: Now that we have the transformed values, we can plot a histogram to see if the data show a Uniform Distribution.  It can be shown that the transformed DataFrame presents a Normal Distribution, so we can take these values to a Linear Regression Model.  Machine Learning works best with numeric values. Let's transform categorical values into numerical values.  In [36]: The method Select\_dtypes (include=['object']) selects the columns from the DataFrame that have an object data type).  In [37]: We’ll use get\_dummies which is a function in Pandas to encode categorical variables.  In [38]: In this case the method concat joins two DataFrames along the columns. This line combines the encoded categorical variables and unencoded numeric variables).  In [39]: Create a dictionary to change 'County\_Ireland' column to 'Ireland' and replace new column name in DataFrame.  In [40]: Change the position of the columns according to the order of the created dictionary.  Linear Regression: "Regression analysis refers to the study of the dependence of a variable, the dependent variable, in relation to one or more variables, the explanatory variables, with the objective of estimating and/or predicting the mean value (of the population) of the first in terms of the known or fixed values (in repeated sampling) of the latter.”  Simple regression: the dependent variable (y) is related to a single explanatory variable (X).  Multiple regression: The dependent variable (y) is related to more than a single explanatory variable (Xi).  In [41]: In this line of code, we identify the variables that will be dependent (y) and independent (X) with ‘y’ being the variable that I am going to predict and ‘y’ the variable that has the characteristics of the variable ‘y’.  In [42]: Display the dependent variable.  In [43]: Import the train\_test\_split class from the model.selection module of the scikit-learn (sklearn) library.  In [44]: Train\_test\_split is a function used to split a dataset into two or more subsets (training and testing), test\_size=0.3 (use 30% of the data for testing and 70% for training), random\_state=42 (set a specific value for random\_state to ensure the results are the same each time the code is run).  In [45]: Import the LinearRegression class from the linear\_model module of the scikit-learn (sklearn) library.  In [46]: Initialize the Linear Regression Model with empty data.  In [47]: X\_train, y\_train will be the variables that represent 70% of the data to be evaluated for the training of the Linear Regression Model.  In [48]: lr.predict predicts 'y' values (output) having another variable, called 'X\_test' (input) as characteristics of 'y'.  In [49]: Display the results of the prediction (variable ‘y’) using a trained Linear Regression Model (lr) on a set of input features (variable ‘X’).  In [50]: np.expm1 is a Numpy function that converts a logarithmic number to an exponential.  Performance metric is used to know the accuracy of the model and compare the predicted with the actual value to know if the model is correct. And for that, we use the r2\_score function as a metric which compares how well the predictions (prediction\_lr) match the actual values (y\_test).  There are also other metrics for regression:  - MSE (Mean Squared Error): average of the squared difference between the real value and the predicted one (penalty for error).  - MAE (Mean Absolute Error): average of the absolute difference (module) between the real value and the predicted one.  In [51]: This code line displays the Determination Coeficient (r2) which has a value of 0.48719343767364476.  Decision Tree Regressor: This model works for Regression and Classification.  Advantages:   * Decision making is more effective and efficient * Ease of use and exhaustive because they review all possible variabilities   Disadvantages:   * They are unstable when the initial data (input) is modified * They tend to overfit (complex model that fits too much to the data instead of generalizing effectively)   In [52]: Let’s import the DecisionTreeRegressor class from the scikit-learn library to create a Decision Tree Regression Model.  In [53]: Create a decision tree model for regression, where the objective is to predict a continuous target variable, random\_state=42 (set a specific value for random\_state to ensure the results are the same each time the code is run), max\_depth=5 (sets the maximum depth of the decision tree to 5).  In [54]: X\_train, y\_train will be the variables that represent 70% of the data to be evaluated for the training of the Decision Tree Regressor Model.  In [55]: dtr.predict predicts 'y' values (output) having another variable, called 'X\_test' (input) as characteristics of 'y'.  In [56]: Display the results of the prediction (variable ‘y’) using a trained Decision Tree Regressor Model (dtr)  In [57]: Yellowbrick is a Python library used to improve the visual diagnosis and model selection process when working with scikit-learn.on a set of input features (variable ‘X’).  In [58]: Import PredictionError from yellowbrick.regressor library to create a graphic that determines the Coefficient of Determination (r2) and identifies how close the line with the predicted values is to the line with the real values.  In [59]: Display the Determination Coeficient (r2) which the relationship between the real values and the predicted values and has a value of 0.5931680852278041.  Random Forest Regressor: this model works with several Decision Tree Models.  For Regression it uses the average of the results of the trees and for Classification a selection is made by the most voted.  Advantages:   * Reduces the occurrence of Overfitting (variance), reduces bias (variance) * It's flexible because we can customize it * Works well with missing and null data * he precision is good because it works with several trees with small depths * It's good for working with Bigdata (more attributes are better)   Disadvantages:   * It has several trees so it tends to be complex (greater decision making when having several trees) * Demands greater computing power, takes longer to process information   In [60]: Import the RandomForestRegressor class from the scikit-learn library to create a Random Forest Regression Model.  In [61]: initializing the model with certain parameters (the idea is to choose the best parameters for the model, called hyperparameters), n\_estimators=10 (set the number of decision trees in the random forest to 10).  In [62]: # X\_train, y\_train will be the variables that represent 70% of the data to be evaluated for the training of the Random Forest Regressor Model  In [63]: # rfr.predict (predicts values 'y' (output) having another variable, called 'X' (input) as characteristics of 'y')  In [64]: Display the results of the prediction (variable ‘y’) using a trained Random Forest Regressor Model (rfr)  In [65]: Generate a graphic that determines the Coefficient of Determination (r2) and identifies how close the line with the predicted values is to the line with the real values.  In [66]: Display the Determination Coeficient (r2) which has a value of 0.6244015686025153.  Performance Techniques are used to optimize the model and obtain the better results. I'll choose the Decision Tree and Random Forest models that showed higher (r2) values.  we can use different optimization tools as shown below:  A) Cross Validation: is a widely used technique to evaluate the performance of models and subsequent verification of trends in machine learning. It consists of partitioning the data into subsets (parts), where one subset is used for training and another subset is used for testing and evaluating the performance of the model  B) Randomized Search: defines a search space as a bounded domain of hyperparameter values and random sample points in that domain" (searches for the best hyperparameters of the model and chooses possible combinations at random).  C) Grid Search: makes all possible combinations between hyperparameters. It takes a longer computational time to obtain the results  Optimize Decision Tree  In [67]: Import KFold class from the scikit-learn library to split the data into different pieces  In [68]: KFold is a cross-validation technique used in which the data set is divided into K folds of equal size, and the model is trained and evaluated K times, n\_splits=5 (split the data into 5 pieces), shuffle=True (choose random numbers)  In [69]: Import cross\_val\_score class from the scikit-learn library.  In [70]: Cross\_val\_score is a function that performs a cross-validation scoring of the estimator (dtr) on the input data (X\_ttrain, y\_train), estimator=dtr (specifies the Decision Tree Regressor Model to use), scoring='r2' (5 (r2) will be obtained since we have 5 partitions and the (r2) will be the average of all).  The average accuracy of r2 is 0.615.  In [71]: Select a series of hyperparameters to then estimate which will be the best of them. Criterion (measure the quality of a split: 'squared\_error', 'absolute\_error'), max\_depth (the maximum depth of the tree ), min\_samples\_leaf (the minimum number of sheets), min\_samples\_split (the minimum number of samples), max\_features (find node split precision: 'sqrt', 'log2', 'auto')  In [72]: Import RandomizedSearchCV class from the scikit-learn library  In [73]: The RandomizedSearchCV is used for hyperparameter tuning by performing a randomized search over the specified hyperparameter values, n\_iter=5 (number of random combinations of hyperparameters to try), cv=cv (cv parameter is a cross-validation splitting), scoring='r2 (use the R-squared score as the evaluation metric), n\_jobs=-1 (low value so that the machine does not collapse).  In [74]: X\_train, y\_train will be the variables that represent 70% of the data. It takes time to process this code as it processes the possible combinations  In [75]: Display the values obtained from the possible combinations made  In [76]:  In [77]:  In [78]:  In [79]:  In [80]:  In [81]:  Optimize Random Forest  In [82]:  In [83]:  In [84]:  In [85]:  In [86]:  In [87]: |