**Section 1: Statistics**

To begin we will interrogate the central tendencies of our data and then summarise the dataset using descriptive statistical methods and some basic plot visualisation.

a) Calculate the central tendency

A table with numbers and a number on it

Description automatically generated

**(Figure 1)**

(i) Getting Mean

A screenshot of a computer

Description automatically generated

**(Figure 2)**

(ii) Getting Mode

Results not relevant due to nature of df

(iii) Getting Median

A screenshot of a computer screen

Description automatically generated

**(Figure 3)**

b) Calculate the variance and standard deviation of features

(i) Variance

A screenshot of a computer

Description automatically generated

**(Figure 4)**

(ii) Standard deviation

A white background with black text

Description automatically generated

(**Figure 5)**

c) Compare and give interpretation to these results.

Histograms –

A graph of different types of data

Description automatically generated with medium confidence

**(Figure 6)**

Boxplots -

A graph with a bar graph and numbers

Description automatically generated with medium confidence

**(Figure 7)**

A graph with a blue rectangle and black squares

Description automatically generated

**(Figure 8)**

A graph with a bar and a line

Description automatically generated with medium confidence

**(Figure 9)**

A graph of a number of people

Description automatically generated

**(Figure 10)**

A graph with a bar graph

Description automatically generated with medium confidence

**(Figure 11)**

A graph of disposable income

Description automatically generated

**(Figure 12)**

We will first need to create a new discrete variable class which will categorise our data into three categories which analyses tourist attractions. This will be done by classifying the three of the ‘tourism’ variables that we have included –

* Total Activities
* Total Accommodation
* Total Attractions

By using the quantile values of central tendency measures, we can calculate how each county ranks in relation to each other and then create discrete categories for each of the values above. This is done calculating the values of each on the 33% and 67% quantiles. Our three new discrete variables are –

• Activities Range (High, Middle, Low)

• Accommodation Range (High, Middle, Low)

• Attractions Range (High, Middle, Low)

Now that we have our discrete variables, we can use binomial distribution to analyse the probability of occurrence within our dataset.

The first problem we will look at is, assessing what the probability of occurrence that a country with a high level of household income will also have a high level of amenities across our three ranges.

We will do this by

* Deciding success parameters
* Calculating our number of test cases (n)
* Calculating the number of outcomes we care about (k)
* Calculating the probability of these outcomes (p)

Our question is to –

Calculate the probability of a County having a high Household Income (million EUR) if it has a High 'Accommodation Range' Using Binomial Probability Mass Function (PMF)

**Question 1:** Calculate the probability of a County having an above average Household Income (million EUR) if it has a High 'Accommodation Range' using Binomial Probability Mass Function (PMF)

k = **3**

n = **26**

p = **0.115384**

A screenshot of a computer

Description automatically generated

**(Figure 13)**

Probability of a County having 'High' accommodation range and an above average Household Income: **0.238105**

**Question 2:** Calculate the probability of a County having a below average Disposable Income(pp) if it has a low 'Activity Range' using Binomial Probability Mass Function (PMF)

k = **7**

n = **26**

p = **0.269230**

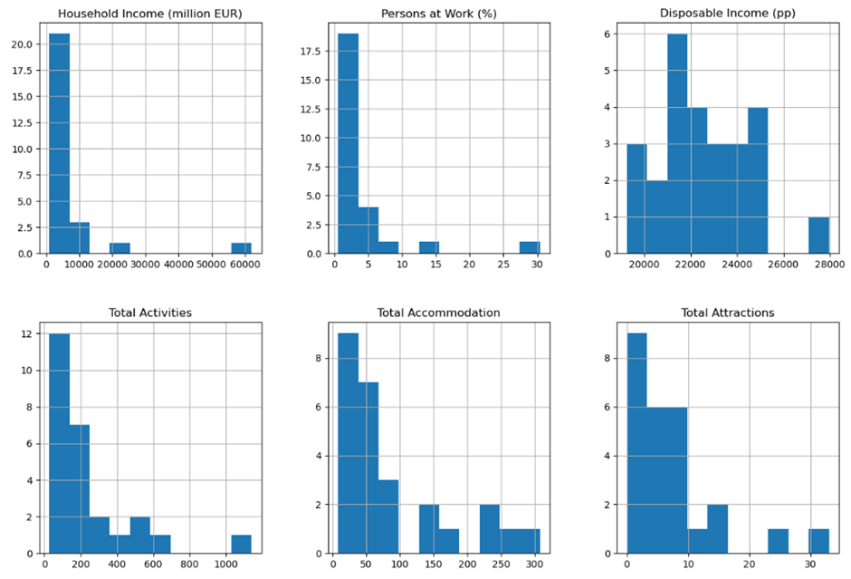
A screenshot of a computer

Description automatically generated

**(Figure 14)**

Probability of a County having 'Low' activity range and a below average Household Income: **0.174097**

First we will need to look at the distribution of our data and see which of our variables have normal or skewed distribution levels.



**(Figure 15)**

From the above we can see that the distribution of all of our 6 variables is skewed, except for one – ‘Disposable Income (pp)’ . Let us take a closer look at this variable and it’s distribution –

A graph of disposable income

Description automatically generated

**(Figure 16)**

We can test this theory on our dataset which looks like it has a standard distribution by formulating questions around this theory and our dataset, we could look at something like the two questions below -

**Question 1:** What is the probability that our Disposable income (pp) is less than €24,000?

First, we can calculate the mean and standard deviation of our desired variable to get our mu and sigma values and then use the scipy norm cdf (cumulative distribution function) to find our answer -

mu = **22626.134615**

sigma = **2028.213264**

A screenshot of a computer code

Description automatically generated

**(Figure 17)**

Answer = **0.750916 %**

**Question 2:** What is the probability that our Disposable income (pp) is between €24,000 and €28,000 (the right-hand side of our graph)?

First we need to calculate the values less than or equal to €28,000 (using the same method as above)

mu= **22626.134615**

sigma= **2028.213264**

A screenshot of a computer

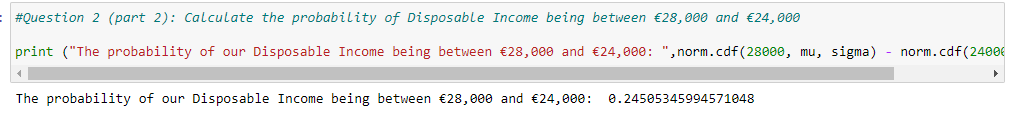
Description automatically generated

**(Figure 18)**

Answer **= 0.995970 %**

Next we need to calculate the difference between our two results by using the following –

norm.cdf (amount 1, mu, sigma) - norm.cdf (amount 2, mu, sigma)



Answer **= 0.245053 %**

The importance of the discrete distributions in our Binomial questions can be highlighted by the fact that these allowed us to analyse elements of our data in a probabilistic manner. The choice of variables for discretization, such as Total Activities, Total Accommodation, and Total Attractions, is justified based on their relevance to the stated problem as these variables capture different elements of the tourism infrastructure in Ireland. By ‘discretizing’ them into their three categories of ‘High Range’ , Middle Range’ and ‘Low Range’ we cand start to ask significant questions about various probabilities which has the potential to give insight into the data that is not obviously apparent or clear from a surface level overview. By grouping these variables, we were enabled to start asking additional questions about the economic factors which might uncover a correlation between features.

It is the authors view that in the context of this project, the discrete nature of variables like Total Activities, Total Accommodation, and Total Attractions made them better suited for binomial analysis, as they represent distinct categories rather than continuous measurements. If the problem being looked at was more focused on the ‘Economic’ features for dependent variables, then a normal distribution approach would have been more appropriate.

**Section 2: Data Preparation and Visualization**

**1. Exploratory Data Analysis:**

This project aims to infer the correlations between common tourist amenities and economic factors specific to local regions (Counties) of Ireland. At the outset of this task, it was clear that several datasets would need to be combined to create data that might offer insights into this problem. This being the case, EDA was utilised across all of these datasets in different ways and for different reasons. Below is a sample of these investigations, the methods that were used and the insights gained through this process.

Characterising the Dataset

When dealing with a new dataset like this, it is important to do some preliminary investigations to gain a better understanding of the type of data that you will be dealing If this is done thoroughly it can give an analyst a better idea of the challenges within the dataset. We can do this by using various methods such as the ones listed below -

shape(): This tells us how many features and entries are contained in our data

head() : This allows us to see a sample of the first rows of the data

info() : This provides a summary of information about the data

describe() : This provides us with the statistical details of the data

nunique(): This helps us to identify if a column is continuous or categorical

**Head:**

A screenshot of a computer

Description automatically generated

**(Figure 19)**

**Shape:**

A screenshot of a computer

Description automatically generated

**(Figure 19)**

**Describe:** By getting the central tendencies and other information from our data we can better understand what other steps may need to be taken for efficient data preparation.

A screenshot of a computer screen

Description automatically generated

**(Figure 19)**

**Info:**

A screenshot of a computer

Description automatically generated

**(Figure 19)**

**Null / NaN values:**

A screenshot of a computer

Description automatically generated

**(Figure 19)**

**Unique Values:**

A screenshot of a computer program

Description automatically generated

**(Figure 19)**

The process above will give us a good idea of the type of data that we will be handling in this project and there is no substitute for these types of investigations, especially when getting familiar with a new dataset (Müller and Guido, 2017).

**Cleaning:**

The first thing that we must do when dealing with a dataset of this size is to clean it. This should be done at the beginning, once you have a general idea of the issues that you may be faced with in the dataset. The data used here was cleaned using the following methods -

**Dropping unneeded columns:**

A white background with red text

Description automatically generated

**(Figure 19)**

**Renaming features for ease of use:**

A screenshot of a computer code

Description automatically generated

**(Figure 19)**

**Data cross validation:**

A screenshot of a computer code

Description automatically generated

**(Figure 19)**

**Merging Data:**

Due to the nature of the problem we are trying to solve, it is necessary to merge several data sets. For this project, elements from the four data sets below will be used –

1. Activities2021
2. Accomdation2021
3. Attractions2021
4. Household\_Income2021

After the target variables were identified, the following methods were used to combine data from these sources –

**Feature summation:**

A screen shot of a computer code

Description automatically generated

**(Figure 19)**

**Array concatenation:**

A computer screen shot of text

Description automatically generated

**(Figure 19)**

**Dataframe Merging:**

A screenshot of a computer code

Description automatically generated

**(Figure 19)**

**2. Preparing data for Machine Learning**

**Elements of Pre-processing and Feature Engineering for ML**

**1. Feature creation**

As mentioned above, several data sets have been combined to create the dataset that is being used within this project. From that, we have also created new features from the data sets that have been used. This has been done in the three following ways –

* Feature summation
* Array concatenation
* Data frame Merging

**2. Feature Selection**

The features that we will focus on for the rest of this project will be the ones created and merged in the section above and they are the following –

* Household Income (million EUR)
* Persons at Work (%)
* Disposable Income (pp)
* Total Activities
* Total Accommodation
* Total Attractions

By having an even split of three ‘Economic’ factor features and three ‘Tourism’ factor features we should have enough scope for the investigation outlined in the original problem.

**3. Imputation**

Imputation is an important process in data preparation, it aims to find missing values within a dataset and then a decision needs to be made by the author as to which method will be used to replace or remove these missing or incomplete values.

Looking at our primary datasets used, we must test for missing values. This was tested by looking at the variable that we are targeting (AddressRegion) and making sure that it contains the same range of value entries (no NaN or nulls) as our total dataset.

1. Activities2021

A screenshot of a computer

Description automatically generated

**(Figure 19)**

2, Accomdation2021

A screenshot of a computer

Description automatically generated

**(Figure 19)**

3. Attractions2021

A screenshot of a computer

Description automatically generated

**(Figure 19)**

Within our dataset, we did encounter empty values for two of the Counties in our ‘Attraction’ feature (Offaly and Monaghan) -

A screenshot of a computer

Description automatically generated

**(Figure 19)**

These were automatically dropped in the process of summing all values (due to 0+0 not returning a value) and because of this absence they needed to be manually entered into our dataset with values equalling to zero.

We have also looked at our ‘Economic’ categories from our dataset in relation to finding outliers that may need to be taken into consideration.

A graph of a number of household income

Description automatically generated

**(Figure 19)**

A graph with a blue rectangular bar

Description automatically generated

**(Figure 19)**

A graph of disposable income

Description automatically generated

**(Figure 19)**

In the first two boxplots above (figure 25 and figure 26) we can see that there is a lot of rightward skew which indicates to us that the skew is positive and that the majority of extreme (higher data points) taper off more slowly. The third boxplot (figure 27) is the most normal distribution of the three with the mean appearing close to the centre of the box and just to the left of the middle of the plot. While we can infer that there are extreme values within the first two plots which are causing this skew, it is not appropriate to our problem to remove or impute these outlying values. As per the testing done previously within the preliminary EDA and the above, no imputation was needed for this project.

**4. Encoding**

As we are preparing our dataset for the application to machine learning models, we will need to convert all the categorical variables that we want to look at into numerical variables. We will apply two types of encoding to our data, ordinal/label encoding and one hot encoding.

**(i) Ordinal Encoding:**

We have three features that we created by finding the statistical ranges of the entries and grouping our entries based on their relative values. These three features are –

* Activity Range
* Accommodation Range
* Attraction Range

Within these we have three values (based on percentiles) of High, Middle and Low. These are ordinal values as they denote if an entry has a specific number of a Tourist amenity, so for this encoding we can use ordinal encoding with High =3, Middle =2 and Low =1.

We can do this by using the Ordinal Encoder from sk learn preprocessing library and fitting it to our target variables (Activity Range, Accommodation Range , Attraction Range) and mapping our new values to the range and replacing the existing values with the mapped ones. This gives us the result below -

A white background with black text

Description automatically generated

**(Figure 19)**

**(ii) One Hot Encoding:**

For our remaining categorical data feature (County) a decision was made to use One-hot encoding to encode because it will allow us to encode data in a way that is non-ordinal, preserve the descriptive parameters of our data and let us handle nominal categories that are fit for machine learning purposes ( ref \* Page 78, Feature Engineering for Machine Learning, 2018.) One of the major setbacks of OHE is the increase in the dimensionality of the dataset but as we have used a targeted approach to create our data frame from combined datasets, we are starting with a dataset that isn’t very large so dimensionality shouldn’t be an issue.

We apply our one-hot encoding by first using the Pandas get-dummies function which will create a new feature to our data frame for each entry contained within our target feature of County and assign a value of True or False where this is present –

A table of text on a white background

Description automatically generated

**(Figure 19)**

Once this has been done, we will then have to convert these True/False values into their binary representatives 1 and 0. We can do that by using the Pandas ‘astype(int)’ function. That will give us the following result –

A screenshot of a number table

Description automatically generated

**(Figure 19)**

Now that our data has been transformed with numerical encoding, we can move on to scaling and get closer to apply machine learning models to our problem.

**3. Scaling**

As we have used OHE to encode our data, we can scale our data afterwards without any negative side affects ( \*ref)

Now we need to assess our data to identify if any scaling might be needed, we can do this by looking at the central tendencies of our features.

A table with numbers and text

Description automatically generated

**(Figure 19)**

From the above (figure 31) we can see that the six features that we are looking at, only ‘Disposable Income (pp)’ has a mean that is close to the 50% percentile mark which indicates that it is likely normally distributed. The other five features all have some degree of variance and as such, are likely skewed and not normally distributed. We can remind ourselves of the distribution to prove this –

A group of blue and white graphs

Description automatically generated

**(Figure 19)**

**Standard Scaler:**

In the first case, we will take our feature that displays normal distribution ‘Disposable Income (pp)’, and apply the SkLearn Preprocessing Standard Scaler to it to create a new feature in our dataframe called ‘Dis\_Income\_Stand’

The standard scaler will centre the new mean of the data at 0 and apply a standard deviation of 1.

**MinMax Scaler:**

Then for our remaining five features (Household Income (million EUR), Persons at Work (%), Disposable Income (pp), Total Activities, Total Accommodation, Total Attractions) that have high levels of rightward skew, we will apply the MinMax scaler to create new corresponding features for each of these.

This will map a new range of values to our features based on 0 being the lowest value in the range and 1 being the highest.

When our data has been transformed it will look like this –

A screenshot of a computer

Description automatically generated

**(Figure 19)**

Although the visual distribution of values will remain the same, we have reduced the dimension of range between our values so that they are in a scale that has a closer proportionality to each other. Our data is now in a form that it can be used for Machine Learning.

**3. Insight through Visualisation**

Through the process of exploratory data analysis and data preprocessing, several key insights that have been gained. These have been documented along the way, but below are the ones that have particular significance or illustrate meaningful insights.

**Grouped Bar chart:**

As a way of looking at multiple amounts plotted on one graph, we can use a grouped bar. This can be useful when we are trying to get a better understanding of multiple features in relation to each other across multiple entries. Below is an example of how these can be used to good effect.

A graph of different colored bars

Description automatically generated

**(Figure 19)**

**Scatter Bubble Plot:**

In the bubble scatter plot visualisation below, we can see a complexity of information being displayed simultaneously. The X and Y relationship of two variables (Total Activities and Total Accommodation) is being shown for each individual ‘County’ in our dataset by the colour of each data plot and the Household Income (Million EUR) is being shown by the size of each bubble. This is a novel way to represent 4 variables at the same time and while the correlation relationship can be valuable, having the size of another variable really adds to what is being conveyed.

A graph with colorful dots

Description automatically generated

**(Figure 19)**

**Heat Map:**

The heatmap is another way to quickly convey the relationship between several variables simultaneously. The colour bar legend on the righthand side is essential for the heatmap to ensure that the viewer is aware of how the feature correlation is mapped.

A screenshot of a computer screen

Description automatically generated

**(Figure 19)**

**Density Plot/Histogram**

Previously in the EDA element of the project we plotted other examples of distribution but below are a number of other ways that these can also be plotted.

A green and black graph

Description automatically generated

**(Figure 19)**

A graph of disposable income

Description automatically generated

**(Figure 19)**

A graph of a graph with a red line and blue dots

Description automatically generated

**(Figure 19)**

**Choropleths:**

A final interesting visualisation that we can look at for our data is the geospatial choropleth plot. This allows us to conceptualise our data for all 26 Counties of the Republic of Ireland at once in a way relative to each other. Like the heatmap, this plot gives the viewer of the weight of the data value based on the intensity of colour. While it is impossible to determine the exact figures we are looking at, it is a great visual aid for broad data conceptualisation.

A map of ireland with different shades of green

Description automatically generatedA map of ireland with different colored areas

Description automatically generated

**(Figure 19) (Figure 19)**

A map of ireland with different colored areas

Description automatically generatedA green map of ireland

Description automatically generated **(Figure 19)** **(Figure 19)**

A map of ireland with a green area

Description automatically generatedA map of ireland with different shades of green

Description automatically generated

**(Figure 19) (Figure 19)**

From the above, we can see that there are many ways that we can visualise the data that we work with in a project like this one. In the next section we will discuss the design and implementation of data visualisation in reference to Edwards Tufts design principles.

**4. Visualization Design and Tufts Principles**

**Question 4.** All design and implementation of your visualizations must be justified and detailed in full., referring to Tufts Principles **[0-20]**

Tufts Six principles of graphical integrity that have been considered and utilised in this project are –

1. The representation of numbers, as physically measured on the surface of the graphic itself, should be directly proportional to the numerical quantities measured.

2. Clear, detailed, and thorough labeling should be used to defeat graphical distortion and ambiguity. Write out explanations of the data on the graphic itself. Label important events in the data

3. Show data variation, not design variation

4. In time-series displays of money, deflated and standardized units of monetary measurement are nearly always better than nominal units.

5. The number of information-carrying (variable) dimensions depicted should not exceed the number of dimensions in the data

6. Graphics must not quote data out of context (page 74).

These have been utilised in the following ways

**Section 3: Machine Learning**

**1. Project Management Framework**

**Part 1:** PM Framework: CRISP-DM

CRISP-DM ( Cross-Industry Standard Process for Data Mining) has been selected as the project management framework for the implementation of this project for the following reasons –

* Structured approach to data collection
* Experimentation and refinement at core of approach
* Iterativie approach to deploying machine learning models

CRISP-DM operates along the distinct phase-based iteration of projects, which is very suitable for this project as the author has been required to satisfy individual component elements from statistics, data preparation and visualisation, machine learning and programming. The data being used for each of these individual parts of the project needs to be constantly updated and re-assessed so this iterative approach has been ideal to manage these considerations. This has allowed the author to produce a clean and structured data set that can now be taken forward for applying to machine learning algorithms to address the problem outlined at the beginning of this project.

The data modelling and feature engineering of this project has been a focused process which has required the author to constantly experiment and re-evaluate at all stages. CRISP-DM has allowed this to be conducted in an efficient manner while allowing the flexibility needed for ongoing testing.

**Part 2:** ML Technique used (supervised, unsupervised, semi-supervised)

The question that we are going to look at for this project is ‘Can we try and predict the 'Total Activities' of a County by the additional features that we have selected using machine learning models?’

As this problem will use continuous data that we already have labelled, we will use supervised machine learning approaches. This problem is a regression task due to the nature of the target variable 'Total Activities', which is continuous. Our main goal will be to develop two machine learning models which can understand the relationships between our selected (independent) features and the target variable. These models aim to produce precise predictions regarding the 'Total Activities' based on the chosen relationships.

**2. Machine Learning Models**

The three machine learning models that we will use to address our problem are:

* Linear Regression
* Random Forest
* Support Vector Regression

We will develop and fit our models with test and train data and select hyperparameters to obtain preliminary results. Once this has been done, we will utilise GridSearchCV to hyper-tune these parameters and then get new results for our models and compare and examine them in a comparative results table.

The metrics that we will use to assess our machine learning models are -

* MSE (Mean squared error)
* MAE (Mean absolute error)
* R-squared (R^2/R2)

MSE, MAE, and R-squared are valuable metrics for evaluating the performance of supervised regression models. They provide different perspectives of a model's accuracy, error, and goodness of fit. This allows for a comprehensive assessment of model performance and is a good way to facilitate relative evaluations of multiple ML models.

**Part 2:** Select Hyperparameters and test

**Linear Regression:** First, we will look at the hyperparameters for our Linear regression model, as Linear Regression is a relatively simple model, there are only the following available

* fit\_intercept=False
* copy\_X=False
* positive=True
* n\_jobs=None

Three of these are True/False so testing can be done very easy with a manual switch and the ‘n\_jobs’ parameter can be defined to how many cores computation will be carried out on to speed up the process. The default parameters for Linear Regression returned the most favourable results in our testing.

A number of numbers and letters

Description automatically generated with medium confidence

**(Figure 19)**

**Random Forest:** The hyperparameters that we will look at for the Random Forest model are the following

* 'n\_estimators': [100, 200, 300],
* 'max\_depth': [None, 10, 20],
* 'min\_samples\_split': [2, 5, 10],
* 'min\_samples\_leaf': [1, 2, 4],
* 'max\_features': ['auto', 'sqrt', 'log2'],
* 'bootstrap': [True, False],
* 'criterion': ['squared\_error', 'absolute\_error']

As this is a complex model and there are so many hyperparamters, it is necessary to use an automated approach to testing for parameter optimisation. More discussion on this is below, but the most favourable results returned from manual testing was the below –

A black text with black text

Description automatically generated with medium confidence

**(Figure 19)**

**Which used the following hyperparameters –**

n\_estimators=200,

max\_depth=10,

min\_samples\_split=4,

max\_features='log2',

**Support Vector Regression:** This model falls between the two previous models that we looked at regarding hyperparameter complexity and the following were used for this project –

* 'kernel': ['linear', 'poly', 'rbf'],
* 'C': [0.1, 1, 10],
* 'gamma': ['scale', 'auto'],
* 'epsilon': [0.1, 0.01, 0.001]

It was discovered that the ‘kernel’:’poly’ parameter was causing the notebook to crash so this was left out of further testing. The most favourable manual testing was returned on the results below –

A black text with black text

Description automatically generated with medium confidence

**(Figure 19)**

**Which used the hyper parameters below –**

kernel='linear'

C=0.1

gamma='auto'

epsilon=0.001

**Part 3:** Use Gridsearch Cross Validation to hypertune parameters

**Using GridSearchCV**

Linear Regression is a simple model and does not utilise GridsearchCV for parameter optimisation.

Random Forest, the results received from GridSearchCV were very positive and reflected the best improvement and final scores of all our models across the three-evaluation metrics. The results below show us the efficiency of the cross validation search in action -

A screenshot of a computer code

Description automatically generated

A number on a white background

Description automatically generated

**(Figure 19)**

Support Vector Regression, GridsearchCV was run on this model for over eight hours and it was still running and did not return a result. Based on the resource intensity of this testing, a decision was made to stop the kernel from running and test the grid parameters on a manual basis to see if much approvement could be found for parameter optimisation. From manual testing, it was discovered that the kernel='poly' hyperparameter was the one causing the issue, so this was removed from the parameter grid and the GridSearchCV was re-ran giving the results below –

A screen shot of a computer

Description automatically generated

A number on a white background

Description automatically generated

**(Figure 19)**

In summary, GridSearchCV is a powerful tool which can remove a lot of the time given over to manual hyperparameter testing. This is evidenced by the results from the Random Forest model above and shows how automating the process of hyperparameter testing can really streamline the efficiency of a suitable machine learning model. However, we have also seen that with the SVR model, this can also be a very resource intensive process and sometimes the results might not be worth the time and processing power required, especially on large data sets.

**3. Machine Learning Model Results**

**Comparative Results Tables**

**Evaluation Metric: R2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Default**  **Tune** | **Manual**  **Tune** | **GridSearchCV**  **Tune** | **Scaled and Optimised** |
| Linear Regression | 0.2057 | 0.1466 | n/a | 0.2057 |
| Random  Forest | **0.3953** | **0.2915** | **0.46147** | **0.4583** |
| Support Vector Regression | -1.4181 | 0.1953 | 0.1551 | -0.2226 |

**Evaluation Metric: MAE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Default**  **Tune** | **Manual**  **Tune** | **GridSearchCV**  **Tune** | **Scaled and Optimised** |
| Linear Regression | 38.7011 | 39.1208 | n/a | 0.0347 |
| Random  Forest | **31.6866** | **31.3138** | **26.4331** | **0.0229** |
| Support Vector Regression | 61.9804 | 37.9999 | 38.3939 | 0.0469 |

**Evaluation Metric: MSE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Default**  **Tune** | **Manual**  **Tune** | **GridSearchCV**  **Tune** | **Scaled and Optimised** |
| Linear Regression | 2020.9854 | 2171.3883 | n/a | 0.0016 |
| Random  Forest | **1538.5821** | 2195.8576 | **1370.3093** | **0.0011** |
| Support Vector Regression | 6153.2259 | **2047.5482** | 2149.7513 | 0.0025 |

**(Table 1)**

**4. Result Interpretation and Findings**

The machine learning analysis of this project focused on predicting the 'Total Activities' of Irish Counties based on selected features using three machine learning models - Linear Regression, Random Forest, and Support Vector Regression (SVR).

For each model, hyperparameters were selected and tested, with GridSearchCV employed for parameter optimization where applicable. The models were then assessed using the following evaluation metrics - Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2).

A graph with green blue and orange bars

Description automatically generated

**(Figure 19)**

**Linear Regression**, being a simple model, did not utilise GridSearchCV. The initial results showed moderate performance across evaluation metrics.

**Random Forest** exhibited significant improvements after parameter tuning using GridSearchCV, with notable enhancements in R-squared and lower error metrics.

**Support Vector Regression** posed challenges due to its resource-intensive nature which meant that manual testing was required when GridSearchCV failed to produce results. Despite these efforts, the model's performance remained suboptimal compared to the other models.

A graph with different colored squares

Description automatically generated

**(Figure 19)**

Our ‘Comparative Results Tables’ (Table 1) present the findings or the machine learning model testing and model performance across the three evaluation metrics. With the improvements of the hyperparameter tuning on model effectiveness very clear. The Random Forest model clearly out preformed the others and demonstrated superior performance both pre and post hyper parameter tuning.

A graph of a bar chart

Description automatically generated with medium confidence

**(Figure 19)**

A graph of different colored squares

Description automatically generated

**(Figure 19)**

In conclusion, while GridSearchCV proved effective in optimizing model parameters, its suitability varied across models. While beneficial for Random Forest, it was less efficient for SVR due to resource constraints. These findings underscore the importance of user/human input into machine learning approaches and make a strong argument for the necessity of analyst supervision. The ‘Comparative Results Tables’ (Table 1) offers high level insights into the relative strengths and weaknesses of each model in the context of this problem and allows for informed decision-making at a model selection and deployment level.

**Section 4: Programming**

As the author is coming from a software development background, Pep8 coding practices were largely adhered to throughout the course of this project. Some considerations for this were the ensuring that meaningful and consistent names were given to variables and functions. Comments were used extensively throughout the project to describe what was being carried out in the many cases where complex actions were being conducted and the variable names could not be descriptive enough. Indentation and spacing was used in a consistent manner to facilitate an easier reader interfacing experience. By taking steps to ensure the basic steps above were implemented congruently throughout the project, it made the programming an easier and more enjoyable experience but more importantly it allows code to be shared for collaboration or a future point of reference.

This project was approached from a pythonic programming perspective. The libraries that were utilised throughout were the following –

**Pandas** - open-source data analysis and manipulation tool.

**NumPy** - package for scientific computing package for and is used in numerical tasks

**SciPy** - library for scientific computing with additional functions for numerical integration, interpolation, optimisation

**Seaborn** - statistical data visualization library based on matplotlib

**Matplotlib** - detailed library for creating static, animated, and interactive visualisations

**Geopandas** - open-source pandas library for geospatial data

Throughout the course of this project, a structured imperative approach was taken to programming paradigms. These two paradigms can be used to work together in the ways in which were required by the author and therefore chosen for this complimentary relationship. Some of the aspects of these paradigms which define this complimentary relationship are through control flow, modular capability for re-use, state management and ease of testing and debugging. Control flow was used in the implementation of writing nested loops, conditional statements, and subroutines. These allowed the author to write code which provide step by step instruction which is easy to understand and can be efficiently organised.

An example of this can be seen in the data visualisation section, where a for loop was used to filter and iterate through the code to be used for a specific visualisation.

A screenshot of a computer code

Description automatically generated

**(Figure 19)**

Another example of how control flow was used was when ‘if else’ statements were nested within a category assigning function which was necessary to answer Binomial questions in the Statistics part of this project.

A computer code with many colorful text

Description automatically generated with medium confidence

**(Figure 19)**

Modularity was used to create functions which broke down complex tasks into more easily understood blocks of code and were more manageable to program. This allowed the author to easily re-use these blocks of code which was essential to the type of comparative approach which was required for machine learning models. Some examples of how this procedural module approach below are below –

A screenshot of a computer program

Description automatically generated

**(Figure 19)**

Being able to define and then re-use the testing and training data for the various machine learning models save the author a lot of time and allowed for more robust manual hyper parameter testing practices to be applied –

A screenshot of a computer program

Description automatically generated

**(Figure 19)**

State management was used to manipulate the data into various forms during the EDA process of the project. This was necessary because of how the data would need to be transformed multiple times to try and draw out new information but the ability to revert to older states of the data was needed to maintain a consistent back stop point as a benchmark for the data. Some examples of this are how the data was transformed into different scaling test features using MinMax and Robust to test different distributions of the data. Another example is how the data was taken from a raw form, merged into different test arrays and data frames, and then eventually encoded using ordinal and one hot encoding to create a multiplicity of data states while always having a handle on how this was being managed.

Due to the complexity of elements to this project (Statistical, Data Visualisation and Machine Learning) Testing and debugging was an integral part of the process from the beginning. A lot of this work is not seen in final products or reports, but it was essential to this project. For the statistical aspect, much testing was required to clean, merge, and transform the data that would eventually be used with the statistical testing. Debugging was used to identify any incorrect data transformation and to resolve unexpected behaviours. For visualisation, this was much the same and really helped to look at how each line of code was behaving and what impact it had on the final visualisations. The structured approach allowed changes to be made at a granular level which meant that once these were resolved, the totality of the visualisations could work as intended.

For machine learning, testing and debugging was essential – one particular case where this proved invaluable was in the application of GridSearchCV to a Support Vector Regression model that was built. GridSearchCV took over 8 hours to run on this model and still had not finished, but because the code had been written in the imperative paradigm, it meant that the author could break down the code into smaller sections to identify where the resources were being consumed in running the cross validation and through this testing the offending hyperparameter was removed from the parameter grid and the GridSearchCV was re-run and returned results in a much less resource intensive manner and allowed for results to be obtained for reporting.

In conclusion, by adopting Pep8 coding practices the author employed a pythonic programming approach to solve multiple problems across three distinct areas of Data Analysis (Statistics, Visualisation and Machine Learning). Through the knowledge gained via lectures and additional research, multiple Python libraries were used to give form to the complexity of ideas which were attempting to be realised by the author. The structured imperative approach allowed for an integration of control flow, modular capability, state management, testing, and debugging throughout the project. The flexibility and power of pythonic programming ensured that this project could be delivered on time and to a standard which hopefully addresses the problems set at the outset.