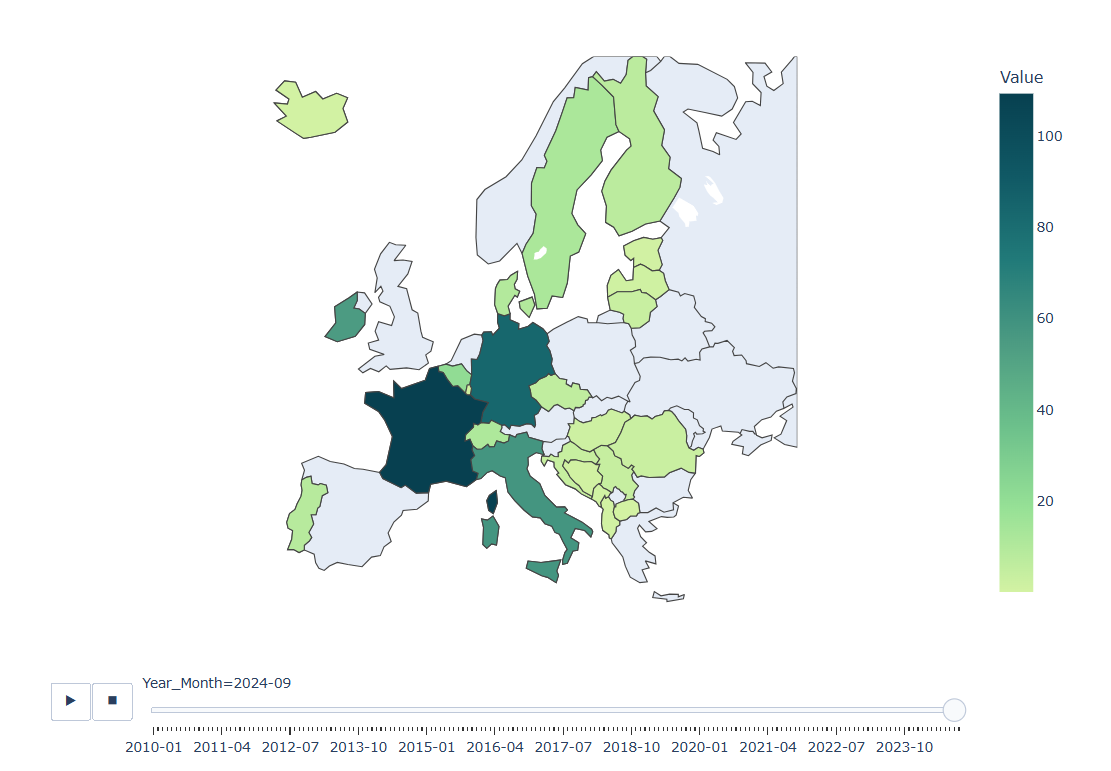
*In this assignment, I will analyze the irish production of bovine meat in slaughterhouses. The aim is to prepare the data, create meaningful visualizations, and compare the production levels with one or more countries in Europe. Using statistical methods, I will explore the differences and similarities between countries, highlighting any significant insights.Additionally, I will apply regression models to identify relationships between variables and use predictive models to forecast future trends in bovine meat production. This approach combines data preparation, visualization, and statistical modeling to better understand the data.*

**Irish Production of Bovine Meat**

The dataset includes the monthly production of bovine meat for all European Union countries, from 2010 to 2024. To make the data more manageable and easier to analyze, it was important to clean and preprocess it. The first step involved checking for any missing data. Additionally, dropping unnecessary columns such as unit, frequency, meat type, and others were removed, as they were not relevant to the analysis.

Since the dataset contains information for all European countries, visualizing the data on a geographic map, such as a choropleth, would be effective. This type of visualization can highlight production differences across the countries. However, to create a choropleth map, the the plotly.express libary requires the ISO country codes for each country. I extracted the ISO codes using pycountry library from the country name column and incorporated them in a new column, ensuring it met the requirements for geographic visualization.



Using the choropleth map, I discovered that a few values were missing from the dataset. These missing values were not flagged when I searched for them. The reason became clear upon closer inspection: for certain months in some countries, there was no information provided at all, meaning the rows for those specific entries were not in the dataset at all.

To fix this, I reorganized the data structure using the pivot command. By pivoting the data, I transformed the monthly entries into index column and each country in a column. It will make easier to spot the missing values, as they now appear as NaN in the corresponding month columns for the affected countries.

To Handle the missing values I tested to different imputation methods:ffill/bfill and KNNImputer. The first one goes through the data back and forward replacing the missing value with the previous value and it was not creating a organic data when seeing in a plot as it was creating a straight line cross when there was a long period of missing data, so I dont see this method as suitable in this casa. The second one estimate the missing value based in their nearest neighbors, generating a more dynamic graph, as I intend to also apply KMean to cluster my data as a next step, this method sees more suitable.

To handle the missing values, I tested two different imputation methods: forward-fill/backward-fill (ffill/bfill) and the KNNImputer. The forward-fill/backward-fill method works by replacing missing values with the last available value before or after it. However, when visualizing the results, this method created straight-line segments across periods with long missing data, it did not look organic or realistic. Because of this, I found it unsuitable for this dataset.

KNNImputer estimates missing values based on their nearest neighbors, resulting in a more dynamic and natural graph. Since my next step involves applying K-Means to cluster the data, the KNNImputer seemed like the more appropriate choice.

To choose against each countries I would compare Ireland, I created groups with similar bovino production countries using 2 different methods: K-means and Hierarchical clustering. K-means divide the data in predefined number of clusters (Collected using Elbow method).Hierarchical Clustering, using the Ward method, builds a hierarchy of clusters by minimizing the variance inside each cluster. I also did K-means cluster using missing value datas.

All of them returned very similar results regarding where Ireland was close, and the closest countries were Spain and Poland.

**Statistical Analysis**

Now that I have identified the countries most similar to Ireland in terms of bovine production, I can proceed with statistical analysis. This will include examining similarities and differences, analyzing the distribution of values, and calculating key statistics such as averages, variances, and trends over time.

Before that, I want to generate a bar chart that summarizes how many times there was a production increase of each month across the selected countries over the years. This will provide an initial visual representation of how production trends vary across months and highlight any consistent seasonal patterns or anomalies.

The bar chart will help in quickly identifying the months with the most significant increases in production, setting the stage for further exploration of the underlying reasons for these trends.

One interesting observation is that there has never been an increase in meat production for December in Ireland since 2010. It would be interesting further investigation to understand the proper reason for this.

To analyze how the mean production of bovine meat differs between the selected countries (Ireland, Poland, and Spain), I Will use One-Way Analysis of Variance (ANOVA) due it works better when comparing more than 2 groups (countries). In another words, if there are any common behavier in the meat production for those countries, like being affect by agricultural policies , regulation, demand etc.

The null hypothesis assumes that there is no significant difference in mean production between the countries, meaning any observed differences are due to random variation.

Anova returned F-statistic: 199.82 and P-value: 2.98e-65

To conduct the test, I used the production data (Value) from each country and calculated the F-statistic and its associated p-value. The F-statistic measures the ratio of variation between the group means to the variation within the groups. A higher F-statistic suggests a greater likelihood of differences between group means, while the p-value is extremely small (much less than the commonly used significance level of 0.05), we can reject the null hypothesis. This means there are significant differences in the mean production of bovine meat in at least one of the country or all of them.

For this reason, I applied the t-test, which allows for pairwise comparisons. This enabled me to compare the mean production of one country against another to confirm which pairs of countries have significant differences in their mean production.

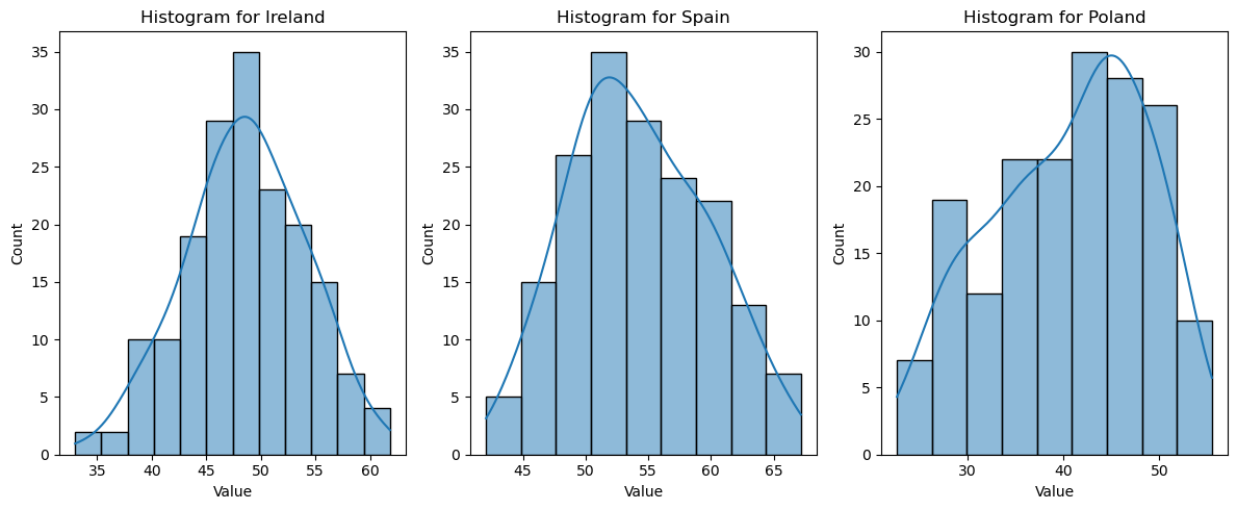
Ireland vs. Spain: T-statistic = -9.59, P-value = 1.74e-19

Ireland vs. Poland: T-statistic = 10.97, P-value = 2.82e-24

Spain vs. Poland: T-statistic = 18.68, P-value = 1.71e-54

All the p-values are significantly smaller than 0.05, which indicates that the differences in mean production between each pair of countries are statistically significant.

To understand more about the data I created histograms graphics to check their distribution of values.



For both Ireland and Spain, the distribution shows in a bell-shaped curve, which means normal distribution. This suggests that the production levels in these countries tend to fluctuate around a central value, with most data points clustered near the mean. In other words, the production values in these countries are relatively stable.

On the other hand, the histogram for Poland does not exhibit the bell-shaped pattern typically associated with a normal distribution. It could be explained due its production increase over the years since 2014, so not cluster near the central value.

However, normal distribution is not particularly relevant in the context of time series analysis, I included it for study purposes only.

Another test applied to the data was Chi-squared which can identify significant association between them, returning:

Chi2-statistic: 159.36287072387887, P-value: 1.0

A p-value of 1.0 is extremely high, meaning there is no evidence to reject the null hypothesis. This suggests that the distributions of production across the countries are independent of one another. One country production doesnt affect the other.

Lets take a better look in Ieland database using 95% Confidence Interval. The confidence interval provides a range of values within which we can be 95% confident that the actual mean production lies. Returning:

Power value: 47.75915

Upper Value: 49.399486

Standard Deviation: 5.513118

Mean: 48.579318

The descriptive statistics for the dataset showed that the mean production level was approximately 48.58, with a standard deviation of 5.51. This indicates that the average monthly production of bovine meat in Ireland during the period has some variability, but remains around this mean value. We can confidently say that the true mean production level lies between 47.76 and 49.40

**Testing Time Series Forecasting Models**

There are numerous models specifically designed for time series forecasting, as well as other models that can be adapted for this purpose. In my previous assignment, I used Facebook Prophet, so, to expand my understanding of different approaches, I will exclude Prophet from this analysis and start by testing the Samirax model.

Samirax is a time series forecasting model designed to analyze sequential data by capturing temporal dependencies and patterns and it requires stationary data. Stationarity means that the statistical properties of the time series like: mean, variance, and autocorrelation, remain constant over time. There are 2 main ways to verify if the data is stationary: by plotting a graph where mean and std lines should follow the data for a stationary result or by applying Augmented Dickey-Fuller (ADF) test. The graphic didnt show the data and means together with std line, confirming the data was not stationary, but I also applied ADF, returning this:

ADF Statistic: -1.4739219717192504

p-value: 0.5463700755837093

Number of Observation used: 161

Critical Value 1%: -3.471633386932248

Critical Value 5%: -2.8796651107461972

Critical Value 10%: -2.576433647235832

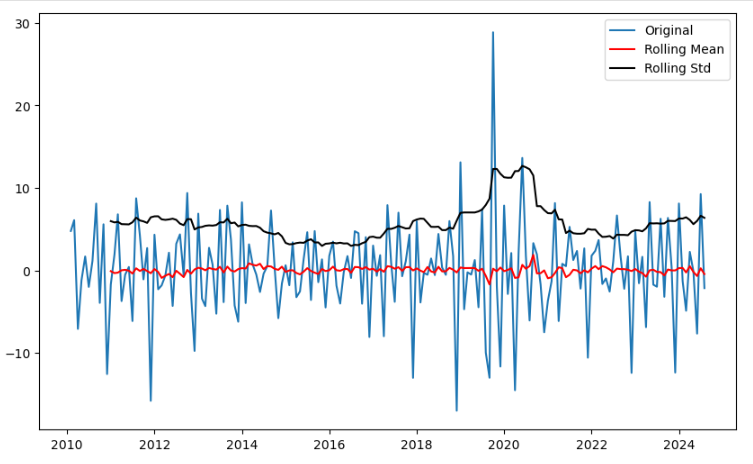
By this result, I can confirm that the data is not stationary because the ADF value is higher than any of the critical values, and the P-value is much higher than 0.05. This means we cannot reject the null hypothesis of non-stationarity.

To address this, we need to apply a method called Differencing, which calculates the difference between the previous value and the future value. Differencing helps to stabilize the mean by removing trends or seasonality, making the data fluctuate around 0. Here's an example of how it works:

Original Data: [100, 105, 102, 108]

Differenced Data: [NaN, 5, -3, 6]

After this, we have our differenced data, as confirmed by the plot and the ADF test:



ADF Statistic: -5.074020786332806

p-value: 1.5807018510073455e-05

Number of Observation used: 160

Critical Value 1%: -3.4718957209472654

Critical Value 5%: -2.8797795410156253

Critical Value 10%: -2.5764947265625

We can now use the differenced data to train our SARIMAX model. For this, we need to split the data into training and testing sets. When working with time series, it is crucial to maintain the chronological order of the data. Unlike standard random splitting methods, we set a specific cutoff date to allocate approximately 80% of the data for training and the remaining 20% for testing.

This approach ensures that the model is trained on past observations and tested on future data, respecting the natural temporal sequence of the time series.

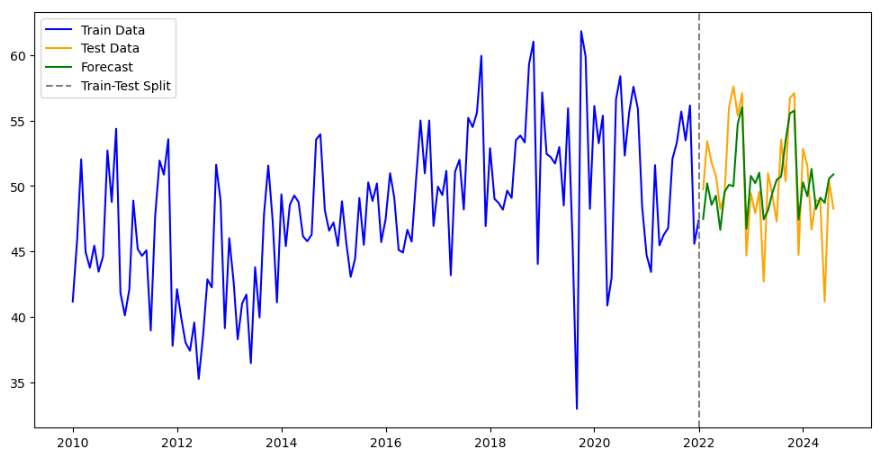
For better performance of the SARIMAX model, it is crucial to identify the best parameters that produce predictions closely matching the test data. This can be accomplished in several ways:

Manual Testing: Manually testing various combinations of parameters, but this method is highly time-consuming and inefficient.

ACF and PACF Plots: Using autocorrelation (ACF) and partial autocorrelation (PACF) plots to guide the selection of AR (AutoRegressive) and MA (Moving Average) terms based on the data patterns.

Automated Approach with auto\_arima: Using the auto\_arima function to automate the parameter selection process, which is not only faster but also more precise. This approach evaluates multiple parameter combinations and selects the configuration that minimizes criteria like AIC (Akaike Information Criterion) or BIC (Bayesian Information Criterion), ensuring an optimal balance between accuracy and model simplicity.

After collecting the best parameter using auto-arima we can create our prediction using our train data. Returning the following prediction:

****

To evaluate the performance of the model, we can use statistical accuracy metrics to compare the predictions against the actual test data. These metrics provide measure of how close the predictions are to the observed values.

Mean Absolute Error: 2.4957261230073824

Mean Squared Error: 9.856990476905553

R² Score: 0.4167731522402941

Mean Absolute error measures the average absolute difference between the predicted and actual values. A lower value indicates that the predictions are closer to the original data. The R² score measures how well the model explains the variance in the data, with a value closer to 1 indicating a better fit.

Auto-Arima is also important to avoid overfitting, by balancing model complexity and fit (using AIC/BIC). Overfitting occurs when a model is too complex, capturing not only the underlying trend and seasonality in the data but also the random noise, leading to poor prediction.

Despite knowing about this I tested the same Samirax approach using less data( from 2015 instead of from 2010).

Mean Absolute Error: 2.71040423403886

Mean Squared Error: 12.70985401689967

R² Score: 0.15662922148752523

The first result (using full data) provided a better score and it is reasonably good, however, we can test other models and try to get a better prediction.

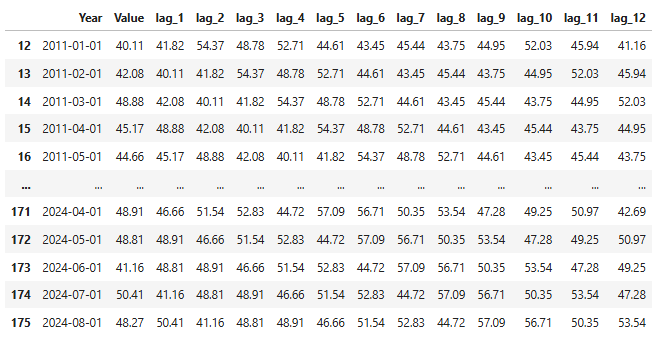
To test different regression methods, I could manually test each one and compare their accuracy results. However, this approach would be time-consuming, especially when working with multiple models and tuning their parameters. To improve this process and improve efficiency, I will use GridSearchCV. This method allows me to automatically search for the best hyperparameters for each model and evaluate their performance based on predefined metrics.

GridSearchCV performs an exhaustive search over a specified parameter grid and selects the best combination of parameters, saving considerable time and effort compared to manual testing.

But before we apply our data in those models we need to look at something else first: In time series forecasting, future values depend on past values. So, to use machine learning techniques, we need to transform this temporal dependency into a supervised learning problem. In other words, we need to reshape the data so that the model can learn from past observations to predict future values.

For example, if you want to predict the current month's production of meat based on the previous 12 months, you create new columns, lag\_1, lag\_2, ..., lag\_12, where each lag column represents the values of the time series from the last 12 months.

This method is a way to make sure the model uses past data to make future predictions.



In the new dataset, the value 40.11, corresponding to 2011-01-01, is derived from the previous 12 months, with each month's value represented in a separate column. Now we separated the lags from the value in X and y respectively, and apply to our GridSearchCV splitting the data using TimeSeriesSplit.

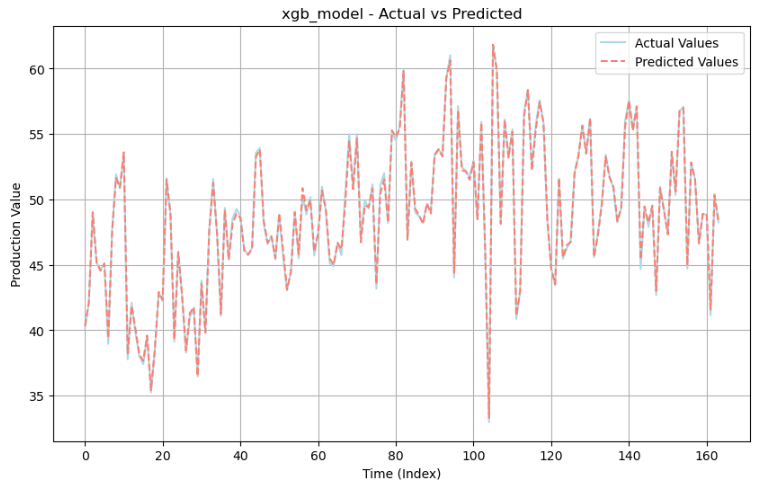
As we already know, it is essential to preserve the chronological order when working with time series data, for this reason, we can not use random split data.

After applying GridSearchCV to get the best predictions for the following regressions: Lasso, Decision tree, Random forest, XGBosster and Neural Network. I got the best accuracy from XGBooster:

Mean Absolute Error: 0.16249046232642175

Mean Squared Error: 0.04865092702512389

R² Score: 0.9984221984384377

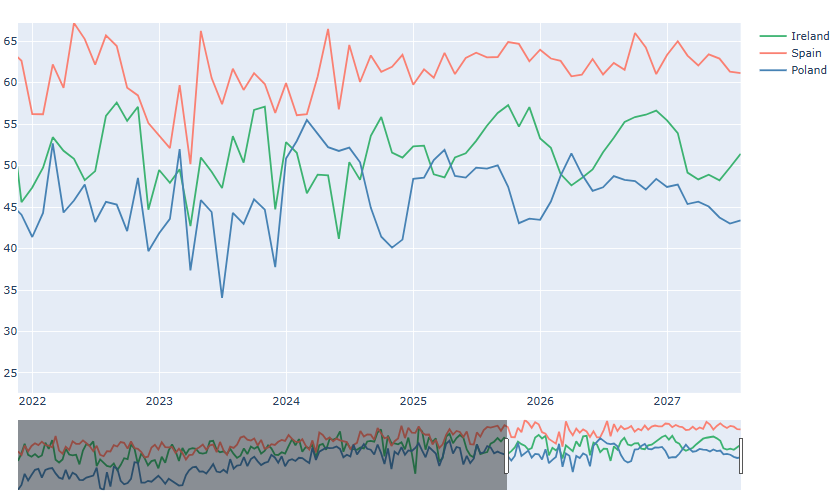


The XGBooster model provided an excellent result not only comparing against the other regression models but also against SAMIRAX model. So I will use it for my forecasting.

**Creating forecast**

After completing all the necessary steps, I am now ready to create forecasts for Ireland, Spain and Poland. The previous stages involved preparing the data, ensuring it was in the right format, and testing different models to find the best one. I used methods like creating lagged features and making sure the data remained in chronological order. After testing several models, the XGBoost model showed the best results, with minimal error and high accuracy.

Now that the model is trained, I can use it to predict future values. By inputting the most recent data into the model, it can generate forecasts based on the patterns it learned during training. This step allows me to make informed predictions about what might happen in the future, based on the historical data we’ve worked with. And this the the plot forecast for 36 months.



The forecast shows a stable and consistent production for Spain, keeping levels between 60-65 tonnes throughout the forecast period.Similarly, Ireland shows a steady trend, but with slightly larger fluctuations, keeping its production levels between 45-55 tonnes.

Poland, on the other hand, is showing a steady upward trajectory, gradually closing the gap with Ireland over the years. While Poland’s growth is promising, the forecast suggests that it is unlikely to surpass Ireland’s production within the next 36 months. However, the narrowing difference signals that Poland could become a closer competitor to Ireland if this growth trend continues beyond the forecast period.

Overall, while Spain remains the leader with consistent and high production, Ireland and Poland are displaying contrasting dynamics, with Ireland maintaining stability and Poland striving to catch up. This trend could have implications for future competition and potential shifts in production rankings over time.

Final prediction plot in Streamlit:

[*https://streamdemo-nhpcwrdfvxlvw6ggpsbu3r.streamlit.app/*](https://streamdemo-nhpcwrdfvxlvw6ggpsbu3r.streamlit.app/)

Sentimental Analyses

Another interesting project I worked on was implementing sentimental analyses in a dataset related to agriculture in Ireland. Sentiment analysis is a subfield of natural language processing (NLP) that involves determining the emotional tone or sentiment in a text. Classifying it as: positive, negative, or neutral sentiment.

Before training a sentiment analysis model, it's crucial to gather a labeled dataset containing both the text and its corresponding sentiment. This data is essential because it provides the necessary information for the model to learn from. In my initial work, I used two datasets: one from Amazon product reviews and another from Twitter. However, my model failed to correctly classify a simple sentence like “I hate this product” as negative. Even when combining the two datasets, the model still struggled to predict correctly. This experience highlighted the importance of selecting high-quality sentiment data for training, as inconsistencies or biases in the data can significantly affect model performance.

After some research, I found the YELP dataset, which is freely available on the YELP website. This dataset contains user comments together with their sentiment scores, making it a valuable resource for training sentiment analysis models.

In this case, the reviews with:

* Ratings of 1 are typically associated with negative sentiment.
* Ratings of 4 or 5 are typically associated with positive sentiment.
* Ratings of 2 or 3 could be classified as neutral, or you could choose to treat them as either positive or negative depending on the range of words used in the reviews.

The only challenge I faced was that the dataset was 5GB in size, which made it take hours to process. To fix this issue, I decided to use the chunksize method, which is a powerful tool in pandas for handling large datasets. By processing the data in smaller parts, I was able to avoid running out of memory and significantly speed up the analysis. Chunking also allows me to work with subsets of the data, which is useful when you only need to process a portion of it. In my case, I used chunking to extract and work with just 1% of the dataset, which is almost 70k comments.

Now we need to preprocess the text data. Text data in its raw form often contains noise, such as punctuation, numbers, and other irrelevant elements that may not contribute to sentiment analysis. For machine learning models to understand and make predictions on the text, we must first use text processing libraries to preprocess it. This will clean our text before turn them into a numerical format that can be interpreted by machine learning algorithms.

We remove these characters to avoid unnecessary noise. For example:

"I love this product!!!" → "I love this product"

"The price of this TV is 199 dollars!" → "The price of this TV is dollars"

"I LOVE this product!" → "i love this product!"

Working → work

Teacher → teach

We combine all the processes above to clear our data.

Once the data is cleaned, we cannot directly vectorize it because machines can only understand numbers, not words or letters. We use the TfidfVectorizer library, which converts the text data into numerical format, enabling the machine learning model to process and analyze it.

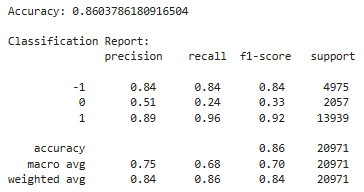
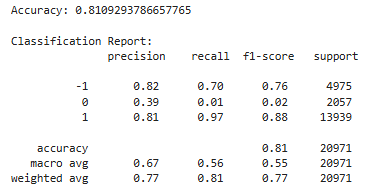
With my data ready to be used in a model, I split it into training and test sets, and applied both Multinomial Naive Bayes (MultinomialNB) and Logistic Regression.

Multinomial Naive Bayes (MultinomialNB) is very common used in text classification tasks, including sentiment analysis. It is based on Bayes' theorem and calculates the probability of each sentiment category based on the occurrence of certain words and assigns the label with the highest probability.

Logistic Regression: also often used in sentiment analysis. It's a linear classifier that calculates the probability that a given input belongs to a certain class (positive, negative, neutral). The model estimates the weights for each feature, and based on these weights, it determines the probability of a review belonging to each sentiment class.

And I got the following accuracy results:

MultinomialNB Logistic Regression



Logistic Regression not only provided the best accuracy in my sentiment analysis task, but it also identified the negative sentiment in the sentence 'I hate this product,' which was something that MultinomialNB failed to recognize. This could be due to the fact that the 1% of the Yeld dataset used might not have been enough to make MultinomialNB efficient, while it was sufficient for Logistic Regression. This highlights that Logistic Regression was better at capturing the nuances of negative sentiment in the text, outperforming MultinomialNB in this case.

With this in my I decided to added one more model, RoBERTa (Robustly Optimized BERT Pretraining Approach). It captures context in both directions of the sentence (left-to-right and right-to-left), it can understand the complex relationships between words in a text. And more important, it doesnt require a labeled sentiment analyses, so avoid training a weak models depending of the labeled data used to train it.

It provides all the libraries to turn the text in vector and analyses it, returning a probability for each possibility (negative, neutral and positive).

This is the result when applying this model to the sentence ‘I hate this product’.

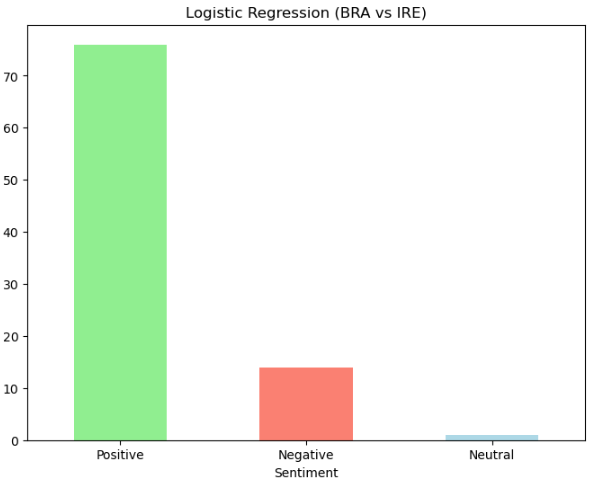
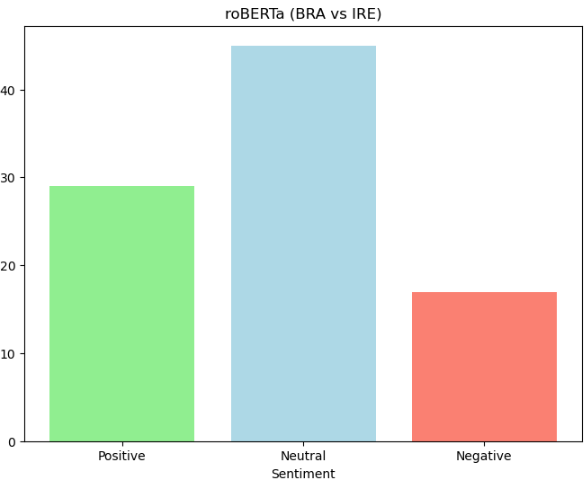
{'negative': 0.9785338, 'neural': 0.01745202, 'positive': 0.0040141526}

An important fact is that RoBERTa calculate all the probabilities and if you added all the values it will return 1. We can see RoBERTa provided a very negative label result. Which is really good.

Now, I can apply real-life data related to Irish agriculture for analysis using both RoBERTa and Logistic Regression.

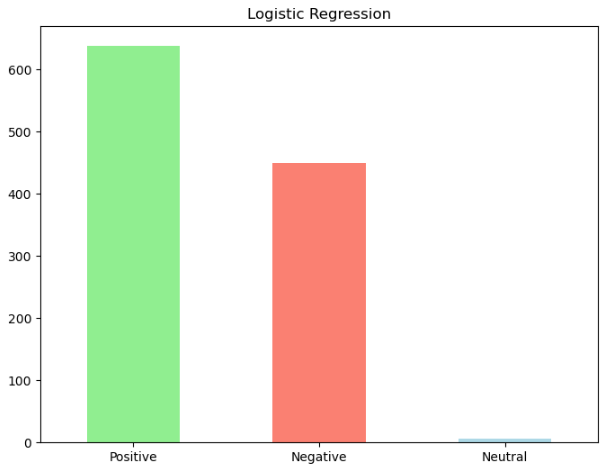
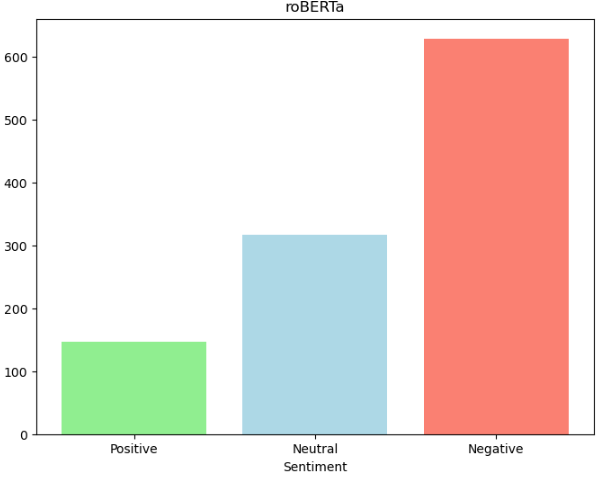
To collect my agriculture data, I used Reddit application to download comments from a forum, so I can apply sentimental analysis to it. I dedicated to test the models in 2 different forums:

The first forum is related to a debate about which country has better meat: Brazil or Ireland.



Based in the comments we can see it is a very reasonable debate and most of the sentences could fit as neutral and we can conclude Logistic Regression fail to understand it and categorized almost all the neutral as positive, in another hand, both manage to categorized almost the same amount of negative sentences, which is very good.

Lets see how the models deal with a more aggressive debate, by importing from Reddit a topic about violence agaisnt irish calves.



Again Logistic Regression fail or showed to be not trained enough to categorized sentences as neutral and probably applying positive category to them. Also not being ‘powerful enough’ to catch many negative sentences as roBERTa did. It could be a result of using only 1% of yeld dataset, showing not be enough to train the model.

### **Final Note**

As part of this analysis, I have taken care to include detailed comments throughout the Jupyter Notebook. These comments serve to explain the important steps of the code, the logic of data processing, and the functionality of the libraries used.

**Database used:**

# *Production of bovine meat in slaughterhouses:* [*https://ec.europa.eu/eurostat/databrowser/product/page/tag00044*](https://ec.europa.eu/eurostat/databrowser/product/page/tag00044)

Reddit:

[*https://www.reddit.com/r/Brazil/comments/1akkad5/brazilians\_in\_ireland\_what\_do\_you\_think\_of\_our/*](https://www.reddit.com/r/Brazil/comments/1akkad5/brazilians_in_ireland_what_do_you_think_of_our/)

[*https://www.reddit.com/r/worldnews/comments/ftpog9/secret\_footage\_shows\_calves\_from\_ireland\_beaten/*](https://www.reddit.com/r/worldnews/comments/ftpog9/secret_footage_shows_calves_from_ireland_beaten/)