**Project NLP | Automated Customers Reviews**

**Executive Summary**

This business case outlines the development of an NLP model to automate the processing of customer feedback for a retail company.

The goal is to evaluate how a traditional ML solutions (NaiveBayes, SVM, RandomForest, etc) compares against a Deep Learning solution (e.g, a Transformer from HuggingFace) when trying to analyse a user review, in terms of its score (positive, negative or neutral).

**Bonus**

The bonus part is to use GenerativeAI to summarize reviews broken down into review score (0-5), and broken down into product categories - if the categories are too many to handle, select a top-K categories.

Create a clickable and dynamic visualization dashboard using a tool like Tableau, Plotly, or any of your choice.

**Problem Statement**

The company receives thousands of text reviews every month, making it challenging to manually categorize and analyze, and visualize them. An automated system can save time, reduce costs, and provide real-time insights into customer sentiment. Automatically classyfing a review as positive, negative or neutral is important, as often:

* Users don't leave a score, along with their review
* Different users cannot be compared (for one user, a 4 might be great, for another user a 4 means "not a 5" and it is actually bad)

**Project goals**

* The ML/AI system should be able to run classification of customers' reviews (the textual content of the reviews) into positive, neutral, or negative.
* You should be able to compare which solution yeilds better results:
  + One that reads the text with a Language Model and classifies into "Positive", "Negative" or "Neutral"
  + One that transforms reviews into tabular data and classifies them using traditional Machine Learning techniques

**BONUS:**

For a product category, create a summary of all reviews broken down by each star or rating (we should have 5 of these). If your system can't handle all products categories, pick a number that you can work with (eg top 10, top 50, Etc)

**Data Collection**

* You may use the publicly available and downsized dataset of Amazon customer reviews from their online marketplace, such as the dataset found [here](https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products/data).
* You also pick any product reviews datasets from [here](https://huggingface.co/datasets/amazon_us_reviews). Make sure your computing resources can handle both your dataset size and the machine learning processes you will follow.

In order to do this, you should transform all the scores with the following logic:

* Scores of 1,2 or 3: Negative
* Scores of 4: Neutral
* Scores of 5: Positive

**Traditional NLP & ML approach**

**1. Data Preprocessing**

**1.1 Data Cleaning**

* Remove special characters, punctuation, and unnecessary whitespace from the text data.
* Convert text to lowercase to ensure consistency in word representations.

**1.2 Tokenization and Lemmatization**

* Tokenize the text data to break it into individual words or tokens.
* Apply lemmatization to reduce words to their base or root form for better feature representation.

**1.3 Vectorization**

* Use techniques such as CountVectorizer or TF-IDF Vectorizer to convert text data into numerical vectors.
* Create a document-term matrix representing the frequency of words in the corpus.

**2. Model Building**

**2.1 Model Selection**

* Explore different machine learning algorithms for text classification, including:
  + Naive Bayes
  + Logistic Regression
  + Support Vector Machines
  + Random Forest
* Evaluate each algorithm's performance using cross-validation and grid search for hyperparameter tuning.

**2.2 Model Training**

* Select the best-performing algorithm based on evaluation metrics such as accuracy, precision, recall, and F1-score.
* Train the selected model on the preprocessed text data.

**3. Model Evaluation**

**3.1 Evaluation Metrics**

* Evaluate the model's performance on a separate test dataset using various evaluation metrics:
  + Accuracy: Percentage of correctly classified instances.
  + Precision: Proportion of true positive predictions among all positive predictions.
  + Recall: Proportion of true positive predictions among all actual positive instances.
  + F1-score: Harmonic mean of precision and recall.
* Calculate confusion matrix to analyze model's performance across different classes.

**3.2 Results**

* Model achieve an accuracy of X% on the test dataset.
* Precision, recall, and F1-score for each class are as follows:
* Class Positive: Precision=X%, Recall=X%, F1-score=X%
* Class Negative: Precision=X%, Recall=X%, F1-score=X%
* Class Neutral: Precision=X%, Recall=X%, F1-score=X%
* Confusion matrix showing table and graphical representations

**Transformer approach (HuggingFace)**

A classification model, (bonus: summarization), and a dashboard are expected in this section.

**1. Data Preprocessing**

**1.1 Data Cleaning and Tokenization**

* Clean and tokenize the customer review data to remove special characters, punctuation, and unnecessary whitespace.
* Apply tokenization using the tokenizer provided by the HuggingFace Transformers to convert text data into input tokens suitable for model input.

**1.2 Data Encoding**

* Encode the tokenize input sequences into numerical IDs using the tokenizer's vocabulary.

**2. Model Building**

**2.1 Model Selection**

* Explore transformer-based models available in the HuggingFace Transformers, potentially:
  + BERT (Bidirectional Encoder Representations from Transformers)
  + RoBERTa (Robustly Optimized BERT Approach)
  + DistilBERT (Lightweight version of BERT)
  + ...
* Selected a pre-trained transformer model suitable for text classification tasks, and justify your choice.
* Share the accuracy using the pre-trained model on your data **without** fine-tuning. This is your base model

**(BONUS) 2.2 Model Fine-Tuning**

* Fine-tuned the selected pre-trained model on the customer review dataset using transfer learning.
* Configured the fine-tuning process by specifying parameters such as batch size, learning rate, and number of training epochs.

**3. Model Evaluation**

**3.1 Evaluation Metrics**

* Evaluate the base model and the fine-tuned model's performance on a separate validation dataset using standard evaluation metrics:
  + Accuracy: Percentage of correctly classified instances.
  + Precision: Proportion of true positive predictions among all positive predictions.
  + Recall: Proportion of true positive predictions among all actual positive instances.
  + F1-score: Harmonic mean of precision and recall.
* Calculate confusion matrix to analyze model's performance across different classes.

**3.2 Results**

* Model achieved an accuracy of X% on the validation dataset.
* Precision, recall, and F1-score for each class are as follows:
* Class Positive: Precision=X%, Recall=X%, F1-score=X%
* Class Negative: Precision=X%, Recall=X%, F1-score=X%
* Class Neutral: Precision=X%, Recall=X%, F1-score=X%
* Confusion matrix

**Deliverables**

* A PDF report documenting the approach, results, and analysis
* Reproducible source code (jupyter notebook or .py files)
* PPT presentation
* Bonus: host your app somewhere so it can be queried by anyone?