**Self Intro:**

Hi,

My name is Nikhitha, I have 2+ years of experience in data science, machine learning, artificial intelligence, I have worked on end to end data science machine learning projects starting from preprocessing, model building, model validation and model deployment. I am good with visualization also.

I have strong knowledge on Python, including libraries such as Pandas, NumPy, Scikit-learn, TensorFlow, and PyTorch as well as expertise in both supervised and unsupervised algorithms like Linear Regression, Logistic Regression, Decision tree, Random Forest, Gradient Boosting, Ada Boost, K-Means

I have strong knowledge on oracle SQL databases, sequel server where I use databases to fetching the data.

I am an actually individual contributor i have also been part of multiple teams.

I am also familiar with project management tools like Jira, Github

Last 6 months I have been working on Gen AI POC project and started upskilling myself while doing the project in Gen AI.

My Day to Day responsibilities are coding in python, solving Data science machine learning projects

**Customer Churn Prediction**

**1.Project Overview:**

Customer churn prediction is about to identify customers who are likely to stop using a product or service (churn) and those who are likely to stay (not churn). By predicting churn, businesses can take steps to retain those customers, such as offering promotions or improving service. This helps increase customer loyalty and revenue. In this project, we’ll go through the entire process: from collecting and preparing the data, to building and training a model. The goal is to create a model that accurately predicts which customers will churn and which will stay and help businesses make data-driven decisions to improve customer retention.

**2.Problem Statement:**

**Objective**: The goal is to predict whether a customer will churn (i.e., stop using the service) based on their characteristics and behaviour.

**Target Variable**: **Churn** (binary classification: 1 = churned, 0 = stayed).

The task involves using historical data to identify patterns and factors associated with customer churn, and using this information to train a predictive model.

Data Collection:

To build an effective churn prediction model, the following types of customer-related data are typically gathered:

* **Demographic Data**: Features like age, gender, and location.
* **Account Information**: Customer's account details such as plan type, start date, and payment method.
* **Usage Data**: Information related to how frequently and in what manner the customer uses the product or service.
* **Customer Support Interactions**: Data related to the number of complaints, support requests, and resolution status.
* **Churn Information**: The target variable indicating whether a customer has churned (1) or not (0).

**3.Data Exploration:**

**Exploratory Data Analysis (EDA)**

* **Data Visualization**: Use visual tools like histograms, box plots, and scatter plots to understand the distribution of features and their relationships with the target variable.
* **Missing Values**: Identify missing data in features and decide on imputation or removal strategies.
* **Outlier Detection**: Detect and handle outliers using techniques such as the Interquartile Range (IQR) or Z-scores.
* **Feature Correlation**: Use a correlation matrix to check for multicollinearity and identify redundant features.

**4.Data Preprocessing**

* **Handling Missing Values**:
  + For numerical features, impute missing values with the **mean** or **median**.
  + For categorical features, impute missing values with the **mode** or a placeholder category like "Unknown."
* **Feature Engineering**:
  + **Tenure**: Calculate the length of time the customer has been with the company.
  + **Total Charges**: Compute the total amount spent by the customer over time.
  + **Service Usage**: Generate features based on customer interactions with the service.
* **Categorical Feature Encoding**:
  + **Label Encoding**: Apply to ordinal categorical features (e.g., payment method).
  + **One-Hot Encoding**: Apply to nominal categorical features (e.g., contract type).
* **Feature Scaling**: Normalize numerical features using **MinMaxScaler** or **StandardScaler** to ensure they are on the same scale, particularly for models like Logistic Regression.

**5.Feature Selection:**

**Feature selection** is crucial to improve model performance, reduce overfitting, and enhance interpretability.

**Chi-Square Test for Feature Selection:**

The Chi-Square test evaluates the relationship between each feature and the target variable (churn or not churn). It helps identify which categorical features are significantly associated with the target.

* **Preprocessing**: Ensure that both the feature and target variables are categorical. If the features are continuous, discretize them into categories (e.g., bins).
* **Apply the Chi-Square Test**: For each categorical feature, calculate the Chi-Square statistic and its corresponding p-value.
  + Features with a p-value less than 0.05 are considered important and should be kept in the model.
  + Features with a high p-value (above 0.05) can be removed as they are less likely to have a significant impact on the target variable.

**6. Model Selection:**

In churn prediction, multiple machine learning models can be employed depending on the data and performance requirements.

**1. Logistic Regression**

* **Use case**: A simple, interpretable model suitable for binary classification tasks.
* **Pros**: Easy to implement, interpretable coefficients.
* **Cons**: Assumes a linear relationship between features and target variable, which might not always be the case.

**2. Decision Tree Classifier**

* **Use case**: Ideal for capturing non-linear relationships in the data with easy interpretability.
* **Pros**: Can handle both categorical and numerical features, and is easy to interpret. It splits the data based on feature values.
* **Cons**: Prone to overfitting if not tuned properly (e.g., without pruning).

**3. Random Forest Classifier**

* **Use case**: An ensemble of decision trees that improves performance by averaging the results to reduce overfitting.
* **Pros**: Handles overfitting better, robust to outliers, performs well with a mix of numerical and categorical features.
* **Cons**: Less interpretable than individual decision trees.

**7.Model Training:**

**Data Splitting:**

* **Train-Test Split**: Split the dataset into 80% for training and 20% for testing to evaluate the model’s performance.
* **Cross-Validation**: Use **k-fold cross-validation** to assess model stability and generalization, typically using 5 or 10 folds.

**Hyperparameter Tuning:**

* **Grid Search**: Exhaustively search for the best combination of hyperparameters (e.g., tree depth in Decision Trees, number of estimators in Random Forest).
* **Random Search**: A more efficient way of exploring the hyperparameter space, especially with a large number of parameters.

**8.Model Evaluation:**

After training the model, evaluate its performance using the following metrics:

* **Accuracy**: The proportion of correctly predicted instances.
* **Precision**: The proportion of positive predictions that were correct.
* **Recall**: The proportion of actual positive instances correctly predicted.
* **F1-Score**: The harmonic mean of precision and recall.
* **AUC-ROC Curve**: Measures the model's ability to distinguish between churned and non-churned customers.

Additionally, use a **confusion matrix** to evaluate the model’s true positives, false positives, true negatives, and false negatives.

**9. Model Deployment**

Once the model is trained and evaluated, the next step is deploying it for real-time predictions.

* **Model Serialization**: Use **Pickle** to save the trained model.
* **API Deployment**: Deploy the model as an API using frameworks like **Flask** or **FastAPI** to make real-time predictions.
* **Integration**: Integrate the model into existing business systems (e.g., CRM) to automatically predict churn for new customers or customers at risk of churning.
* **Monitoring**: Continuously monitor the model’s performance and retrain it periodically to ensure it stays relevant as customer behaviour evolves.

**Project Explanation:**

1. The main goal of the project is to predict which are all customers are likely to churn, The data was given by the Client.
2. Next, we cleaned the data by handling missing values, converting categorical data into numerical values, and scaling the features to ensure the model would perform well. Then we also detected and dealt with outliers and created new features that might help with the churn prediction.
3. After cleaning the data, we explored it to understand the relationships between different factors and churn and identified important features and used visualizations to recognize patterns and trends in the data.
4. Then we built the model using Logistic Regression, Decision Tree and Random Forest and we split the data into training and test sets and evaluated the models using metrics like accuracy, precision, recall, and F1-score and fine-tuned the models to improve their performance. We got good results with Random Forest model.
5. Then the final model was saved as a pickle file and deployed with flask for predictions.
6. Finally, we used the best-performing model to segment customers based on their churn risk and provided actionable insights for targeting high-risk customers with retention strategies (plans like feedbacks, surveys).

**10 Features**

1. customer\_tenure
2. monthly\_revenue
3. customer\_age
4. plan\_type
5. billing\_method
6. product\_type
7. account\_type
8. churn\_label
9. discount\_usage
10. last\_purchase\_date

**Text Summarization and Document Management:**

**1.Objective:**

The goal of this project is to develop a system that automatically summarizes documents into brief, easy-to-read formats and classifies them into predefined categories. This helps automate document organization, improve information retrieval, and make content easier to understand.

**1.Data Collection:**

* We started by collecting a large dataset of documents from various websites using **web scraping techniques**.

**2.Data Preprocessing:**

* Next, we cleaned the collected data to remove any irrelevant text and ensure it’s in a usable form. For this, we applied standard **NLP preprocessing techniques**, which include:
  + **Lowercasing**: Converting all text to lowercase to maintain uniformity.
  + **Removing Noise**: This involved eliminating special characters, symbols, punctuation, etc., which don't contribute to the text’s meaning.
  + **Stopword Removal**: We removed common words (e.g., "the", "is", "in") that don’t provide useful information for our task.
  + **Normalization**: We used techniques like **stemming** and **lemmatization** to reduce words to their root form (e.g., "running" becomes "run").

**3.Tokenization:**

* We then divided the document into individual tokens (words or phrases). Tokenization helps break down the text into smaller chunks that are easier to process and analyze.

**4.Sentiment Labeling:**

* To add an extra layer of meaning to the documents, we labeled the documents with sentiment tags such as **positive**, **negative**, or **neutral**. This allows the system to understand the sentiment behind the content, which is valuable for certain applications like document classification.

**5.Data Splitting:**

* We split the dataset into two parts: **training** and **testing** data. The training data was used to train the model, and the testing data was used to evaluate the model’s performance.

**6.Text Vectorization:**

* We used **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert the text into numerical vectors. TF-IDF is a statistical method that helps capture the importance of words in the document and makes it easier for machine learning models to understand the text.

**7.Model Selection and Comparison:**

* We tested several machine learning models to find the best one for our task. The models we experimented with include:
  + **Decision Tree**
  + **Random Forest**
  + **Linear Regression**
* We compared the performance of these models using key metrics like:
  + **Accuracy**
  + **Precision**
  + **Recall**
  + **F1 Score**

**8.Model Training:**

* After selecting the model that performed the best during comparison, we trained the model using the labeled training data. This step allows the model to learn patterns and make predictions.

**9.Model Evaluation:**

* Once the model was trained, we evaluated its performance on the testing dataset. We used predefined metrics such as accuracy, precision, recall, and F1 score to assess how well the model generalized to unseen data.

**10.Confusion Matrix Analysis**:

* Finally, we analyzed the **confusion matrix**, which helps us understand the model’s performance in more detail. It shows the counts of true positives, false positives, true negatives, and false negatives, allowing us to identify where the model is making errors and where improvements are needed.

**AI- Enhanced Document Retrieval System:**

**1.Objective:**

The goal of this project is to develop an AI-powered document retrieval system that efficiently extracts and delivers the most relevant responses to user queries. The data is in PDF Format.

**2.Load and Extract Document Text**

* Extract raw text from documents (e.g., PDF) using libraries like **PyPDF2**, **pdfminer**.
* Preprocess the extracted text to prepare it for chunking and embedding.

**3. Split Document into Chunks**

* Divide the document into smaller sections, such as paragraphs or sentences, which will help improve the relevance and accuracy of document retrieval.

**4.Data Preprocessing**

Clean and preprocess the text to improve the quality of the document chunks:

* **Lowercasing**: Convert all text to lowercase to standardize it, ensuring uniformity across the text and reducing complexity.
* **Noise Removal**: Remove special characters, symbols, and punctuation that do not contribute to the meaning of the content and may introduce noise for model processing.

**5.Generate Embeddings Using OpenAI**

* Convert each document chunk into **dense vector representations** using **OpenAI’s embeddings** or similar embedding models like **Sentence-BERT**.
* These vector embeddings capture the **semantic meaning** of the document chunks and facilitate efficient retrieval.

**6. Store Embeddings in a Vector Database**

* Store the generated embeddings in a **vector database** (such as **FAISS**, **Pinecone**, or **Milvus**). The vector database enables fast similarity searches to retrieve the most relevant document chunks based on the query.
* Ensure that the embeddings are indexed for quick retrieval when a user query is entered.

**7. Develop a Web Interface for User Input**

* Create a simple web interface using **Flask** or **FastAPI** where users can enter queries in natural language.
* The interface should allow seamless interaction and send the user's query to the backend for processing.

**8. Process User Query**

* Once the user submits a query, convert it into an embedding using the same embedding model that was used for the document chunks.
* The query embedding is generated by passing the user query through the **OpenAI embedding model** (or an equivalent model).

**9. Query Embedding Similarity Search**

* Perform a **similarity search** by comparing the query embedding with the embeddings of the document chunks stored in the vector database.
* Use **cosine similarity** or **Euclidean distance** to identify the most similar document chunks based on the query.

**10. Ranking Retrieved Results**

* Rank the retrieved document chunks based on their similarity to the query. The top-ranked document chunks are considered the most relevant.
* This ranking ensures that the most relevant documents appear first, providing more accurate context for the response.

**11. Retrieval-Augmented Generation (RAG)**

* **Retrieval Step**: Use the top-ranked document chunks retrieved in the previous step as context for the generative model.
* **Generative Model**: Pass the ranked document chunks along with the original query to a **Large Language Model (LLM)**, such as **GPT-3.5**
  + The LLM generates a coherent and contextually accurate response based on the relevant information in the document chunks.
  + The system uses **RAG** to augment the generative model’s output, ensuring the response is well-informed and contextually appropriate.

**12. Final Response Generation**

* The LLM synthesizes the retrieved chunks and the user query to generate a final, detailed response.
* The generated response is then returned to the user through the web interface, providing a comprehensive and contextually relevant answer.