Self-Introduction

Hi my name is K. Manish royal. I have 2w years of work experience as a data science, machine learning Engineer. currently I am working in APPLITECH Solutions PVT LTD as a data scientist, past 1+ years I have been working here. I have good knowledge in building machine learning projects. Starting from problem definition, preprocessing, model building, model validation to till model deployment.My expertise lies in leveraging libraries like Pandas, NumPy, Scikit-Learn, NLTK, Keras, and TensorFlow and tools like Jupyter Notebook, and VS Code. I have been part of different phases of machine learning projects, supervised and unsupervised Machine Learning Algorithms. Like decision Tree, Random Forest, Linear Regression, KNN, Logistic etc. I have strong knowledge on Python, SQL, Keras, Tensorflow. I am good with visualization also. and experience with libraries such as matplotlib, seaborn. I have a strong knowledge on Oracle SQL database, SQL server, where I use the databases for fetching the data. And also, I am good with NLP Concepts, my day-to day responsibilities involve primarily coding in Python, solving data science, machine learning projects. I am actually an individual contributor. I have also been part of multiple teams. And I have capability to handle the end to end projects. currently I am working on a POC (proof of concept) related GEN AI project. (poc - it's a chatbot which you're trying to build by using open AI.) and I started upskilling myself while doing the project in Gen AI. This is all about me.

PROJECTS

1.AI Chatbots for Personalized Retail Customer Support

Problem Statement

The client previously relied on a manual customer service system to respond to product reviews, which was time-consuming and inconsistent. This approach often led to delays in addressing customer feedback, missed opportunities to engage with customers, and inadequate resolution of concerns. To address these limitations, the client seeks an automated solution to improve efficiency and customer satisfaction.

Objective:

To develop a Generative AI-powered chatbot that automatically responds to product reviews in real time, leveraging customer feedback data such as review sentiments (positive,

negative, neutral), product ratings, common issues, and FAQs. The chatbot aims to deliver personalized, timely responses that acknowledge positive feedback, resolve negative concerns, and address frequently raised questions, ultimately enhancing customer engagement and brand reputation.

1: Install All the Required Packages-- langchain, pypdf, openai.

2. Import All the Required Libraries – PyPDFLoader, OpenAIEmbeddings.

3. Load the Documents and Extract Text from Them.

Using PyPDF we will extract the text from documents.

4. Split the Document into Chunks.

To split a document into chunks, you need to determine the desired size for each chunk, whether based on words, sentences, paragraphs, or pages. Once the document is loaded, split the text into chunks accordingly. Optionally, perform any necessary processing on each chunk, such as text preprocessing or feature extraction. Finally, store or use the chunks as needed for further analysis or processing in your application.

We will split the Documents into chunks using ‘CharacterTextSplitter” from ‘langchain’ to split the extracted text into chunks.

After splitting the document into chunks, each chunk undergoes a series of preprocessing steps,

Preprocessing steps:

1. **Removing Stop Words**: This involves eliminating common words such as "the," "is," and "and" from the text. These words typically do not carry significant meaning and can be safely removed to reduce noise in the data.
2. **Tokenization**: Tokenization breaks down the text into individual words or tokens. This step is essential for further analysis, as it provides a structured representation of the text that can be processed more efficiently.
3. **Converting All Text to Lowercase**: By converting all text to lowercase, we ensure consistency in the data. This prevents the same words from being treated differently due to differences in capitalization, improving the accuracy of subsequent analyses.
4. **Removing URLs**: URLs often appear as noise in text data and do not contribute to the semantic meaning of the content. Removing them helps streamline the data and focus on relevant information.
5. **Removing Email Addresses**: Similar to URLs, email addresses are typically irrelevant for text analysis tasks and can be safely removed to reduce clutter and improve focus on meaningful content.
6. **Converting Emojis to Text**: Emojis are graphical representations used to convey emotions or sentiments. Converting them to text equivalents ensures uniformity in the data and facilitates analysis by treating emojis as regular textual content.

7. **Expanding Text Contractions**: Text contractions such as "can't" or "won't" are expanded into their full forms ("cannot" and "will not," respectively) to ensure consistency in representation and facilitate accurate analysis.
8. **Lemmatization**: Lemmatization involves reducing words to their base or dictionary forms (lemmas). This step helps standardize variations of words and reduces the vocabulary size, leading to more effective analysis.
9. **Removing Punctuation**: Punctuation marks such as periods, commas, and exclamation points are removed from the text. While punctuation serves grammatical purposes, it often does not contribute substantially to the semantic meaning of the text and can be safely eliminated.

5. We Download the Embeddings from OpenAI Embeddings.

We are using text Embedding 3 small as our embedding model because this is the basic one and our project is still in POC

6. Setting Up Chroma DB for storage purposes as our Vector Database.

Embeddings that we got above are stored in Vector Database which is Chroma DB

7. We will create an OPENAPI KEY and then we will use the OPENAPI KEY to access the Model, we will instruct the model by giving the necessary prompt

Here we are using chatgpt 3.5 turbo, and chatgpt4 as our models. We will do prompt enrichment if our answers are not correct, and we can use RAG to get proper and accurate answers.

8)After the prompt, it will convert into embeddings and then based on embeddings it will use semantic search and then we will check whether the answers are correct or not

Here we will use a cosine similarity search to find the distance between words.

9)We have created a memory object that is necessary to track inputs/outputs and hold a conversation: To create a memory object for monitoring inputs and outputs in your chatbot conversation.

We follow the below steps for creating a memory object:

1. Define Memory Structure: Determine the structure of the memory object, including fields to store user inputs, chatbot responses, timestamps, and any other relevant information.

2. Implement Memory Functionality: Write code to instantiate the memory object and add methods for storing user inputs and chatbot responses and retrieving conversation history and timestamps.
3. Integrate Memory with Chatbot: Integrate the memory functionality into your chatbot application, ensuring that user inputs and chatbot responses are properly recorded and stored in the memory object during the conversation.
4. Ensure Data Privacy and Security: Implement measures to ensure the privacy and security of the data stored in the memory object, such as encryption, access controls, and data anonymization.
5. Test Memory Functionality: Test the memory functionality thoroughly to ensure that it accurately tracks conversation history and timestamps and that the data is stored and

---

2. Smart Surveillance System for Intrusion Detection in Retail Stores

Problem Statement

Retail stores face significant challenges in preventing unauthorized access, theft, and other security breaches. Traditional surveillance systems often lack intelligent features to detect unusual activities in real time, relying heavily on manual monitoring by security personnel. This increases the likelihood of delayed responses and operational inefficiencies.

The **problem** is to design a system that can:

1. **Automatically monitor video feeds** for suspicious activities such as unauthorized access, theft, or loitering.
2. **Provide real-time alerts** to security teams for immediate intervention.
3. Be scalable for deployment across multiple retail locations without compromising performance.

Solution

The **Smart Surveillance System for Intrusion Detection** provides an intelligent video monitoring solution that leverages computer vision techniques to detect and respond to security incidents in real-time. The system uses **YOLO (You Only Look Once)** for object detection, integrates with real-time alert mechanisms, and offers an API interface for seamless deployment and scalability.

Preprocessing Steps

1. Video Frame Extraction

- Extract frames from video feeds at intervals (e.g., 1 frame per second) to reduce processing overhead while retaining relevant data.

2. Data Annotation

- Annotate each frame with bounding boxes and class labels:
    - Objects: People, products, etc.
    - Activities: Normal vs. suspicious behaviors (e.g., theft, loitering).
- Tools: Use labeling tools like LabelImg or CVAT for efficient annotation.

3. Resizing and Normalization

- Resize images to YOLO-compatible dimensions (e.g., 416x416 or 640x640).
- Normalize pixel values to a range of [0, 1] for faster and more efficient processing.

4. Data Augmentation

- Improve generalization by augmenting the dataset:
    - Flip images horizontally.
    - Add noise or blur to simulate various lighting conditions or camera qualities.

    - Random cropping to focus on different parts of the frame.

5. Splitting Data

- Split annotated data into:
    - **Training set**: ~70% of the data.
    - **Validation set**: ~20% for hyperparameter tuning.
    - **Testing set**: ~10% to evaluate final performance.

6. Format Conversion

- Convert annotations to the format required by YOLO (e.g., .txt files with class labels and bounding box coordinates).

Technical Approach and Implementation

1. Object Detection using YOLO (You Only Look Once):

- **YOLO Framework:** This system utilized the YOLO (You Only Look Once) model for real-time object detection. YOLO is a popular deep learning model for object detection due to its speed and accuracy, making it suitable for real-time video surveillance.
- **Purpose of YOLO:** YOLO's role was to identify various objects in the store environment (e.g., people, store products, etc.). This allowed the system to recognize suspicious movements, such as unauthorized access to high-risk areas or activities indicating potential theft.

2. Video Processing with OpenCV:

- **OpenCV Integration:** OpenCV (Open Source Computer Vision Library) was integrated into the system to handle video feed processing, helping analyze live video streams from surveillance cameras installed in key areas of the store.
- **Functionality:** OpenCV allowed the system to break down video feeds into frames, which were then analyzed by the YOLO model to detect objects and determine if any activity was unusual or unauthorized.

- **Key Areas of Surveillance:** The system focused particularly on high-risk areas like inventory storage (where expensive products are kept) and cashier points (where money handling occurs).

3. Enhanced Scalability with TensorFlow:

- **Use of TensorFlow:** TensorFlow was employed to enhance the performance and scalability of the detection model. By deploying the system on TensorFlow, it became easier to manage and scale the system across multiple retail stores.
- **Benefits for Larger Retail Chains:** TensorFlow allowed the model to handle multiple video feeds and increase processing speed, making it viable for larger retail chains that operate several stores with high foot traffic.

4. Real-Time Alert Notifications:

- **Alert Mechanism:** To ensure immediate response, the system was configured to send real-time alerts to store security teams via SMS and email.
- **Triggering Alerts:** Whenever the system detected suspicious activity or unauthorized access, it automatically generated an alert and sent it to designated personnel, allowing them to respond immediately.
- **Advantages:** This feature minimized the delay in human response, significantly improving the security team's ability to prevent theft or unauthorized access.

5. Performance Evaluation:

- **Performance Metrics:** The system was tested in real retail environments to evaluate detection accuracy and speed.

Result: By carefully analyzing system performance and making necessary adjustments, the system achieved high accuracy and quick response times, helping reduce theft and losses effectively.

Tools & Techniques

- **Frameworks & Libraries:**
    - **YOLO:** Used for efficient, real-time object detection.

    - **OpenCV:** Facilitated video processing to extract frames from live feeds.
    - **TensorFlow:** Improved the system's scalability and performance, particularly useful for deployment in large retail chains.

- **Languages:**
    - **Python:** The primary language used for implementing computer vision techniques, integrating different libraries, and managing data.
- **Computer Vision Techniques:**
    - **Real-time Object Detection:** Enabled the identification of people, objects, and suspicious movements in video feeds.
    - **Video Processing:** Processed live video streams to analyze frames and detect abnormal behaviors.
- **Other Tools:**
    - **Pandas and NumPy:** Used for handling data related to video feeds, alerts, and detection results, assisting in performance evaluation and system optimization.

3.Fraud Detection in Retail Transactions (Internship project)

Project Overview

This project involved building a real-time fraud detection system designed specifically for retail transactions. The goal was to minimize financial losses by identifying suspicious transactions through advanced machine learning techniques. By analyzing purchasing patterns, unusual spending behavior, and

other transaction details, the system detects anomalies that might indicate fraud, helping retailers safeguard against financial risks.

Goals and Objectives

- **Real-Time Fraud Detection:** Develop a system that can analyze retail transactions in real-time and flag any unusual activity.
- **Minimize Financial Losses:** By catching potentially fraudulent transactions as they occur, the system helps prevent financial losses due to fraudulent purchases.
- **Reduce False Positives:** Ensuring that legitimate transactions aren't flagged as fraud to maintain a smooth customer experience.

Preprocessing Steps

1. Data Cleaning

- Handle missing values:
  - Drop rows or fill with statistical imputation (mean, median, mode).
- Remove duplicate transactions to avoid skewing results.
- Normalize data (e.g., transaction amounts) to standardize input for models.

2. Feature Extraction

Time-based Features:

- Time of day (e.g., midnight transactions may indicate fraud).
- Day of the week or seasonal trends.

Customer Behavior:

- Average transaction amount.
- Typical transaction frequency.

Anomalous Patterns:

- Compare transaction location with customer's historical data.
- Unusually high transaction amounts.

3. Encoding Categorical Data

Convert categorical fields like payment method, location, or customer type using:

- One-hot encoding for non-ordinal categories.
- Label encoding if categories have a meaningful order.

4. Handling Imbalanced Data

- Fraud cases are typically rare compared to legitimate transactions.
- Techniques to address imbalance:
  - Resample the data (e.g., oversample fraud cases using SMOTE).
  - Assign higher weights to the fraud class during model training.

5. Outlier Detection

- Identify and treat outliers in features like transaction amounts to avoid skewed results.
- Use statistical methods (e.g., z-scores, IQR) or pre-trained anomaly detection algorithms.

6. Data Scaling

- Scale numerical data using techniques like MinMaxScaler or StandardScaler for uniformity.

Technical Approach and Implementation

1. Data Preprocessing and Feature Extraction:

- **Transaction Data Analysis:** The system began with analyzing transaction data, including information like purchase amount, location, time, and frequency.
- **Feature Extraction:** Key features were derived from this data to improve fraud detection accuracy. Examples include:
    - **Unusual Spending Patterns:** Transactions with unusually high amounts compared to a customer's normal spending.
    - **Location Discrepancies:** Transactions made in locations that differ significantly from the customer's usual buying areas.
    - **Large Transaction Volumes:** High-value purchases that deviate from the customer's historical patterns.
- **Data Cleaning and Preparation:** The transaction data was preprocessed to remove inconsistencies, handle missing values, and format it for machine learning models.

2. Model Selection and Implementation:

- **Machine Learning Models:** Several models were chosen and trained to classify transactions as either fraudulent or legitimate. These included:
    - **Random Forest:** A versatile, ensemble-based model known for handling complex datasets well. Random Forest works by creating multiple decision trees and averaging their outputs.
    - **XGBoost (Extreme Gradient Boosting):** Known for its high performance and efficiency, XGBoost improves accuracy by iteratively boosting decision tree models, learning from errors in each iteration.
    - **Isolation Forest:** A specialized algorithm for anomaly detection, Isolation Forest excels at identifying outliers, which is particularly useful for spotting unusual transactions.

- **Model Training:** These models were trained on historical transaction data, with fraudulent transactions labeled to teach the models patterns commonly associated with fraud.

3. Feature Engineering:

- **Purpose of Feature Engineering:** To improve the accuracy of fraud detection by identifying and enhancing relevant data patterns.
- **Techniques Used:** Created additional features and refined existing ones to help the models differentiate between normal and suspicious behavior. For instance, derived features that compare current transaction data with historical behavior.
- **Goal:** Minimize false positives (legitimate transactions mistakenly flagged as fraud) and increase precision, focusing on transactions that are highly likely to be fraudulent.

4. Integration into Retail Transaction Systems:

- **System Integration:** The fraud detection models were integrated directly into the retailer's transaction processing system, allowing them to analyze transactions in real time.
- **Monitoring & Flagging:** Once integrated, the system monitored all incoming transactions and flagged any that showed signs of fraud. This enabled immediate action from security teams or further verification with the customer.
- **Real-Time Processing:** By using optimized models and efficient data handling, the system could analyze each transaction in milliseconds, ensuring that fraud detection didn’t slow down the transaction process.

5. Performance Analysis and Optimization:

- **Evaluation Metrics:** The models’ performance was evaluated using precision and recall, two metrics that are crucial in fraud detection:
    - **Precision:** Measures how many flagged transactions were actually fraudulent, reducing false alarms.
    - **Recall:** Measures the model’s ability to catch as many fraudulent transactions as possible, minimizing missed cases.

- **Model Optimization:** The models were tuned to maximize both precision and recall, balancing accuracy with minimal false positives to provide reliable fraud detection without impacting customer experience.
- **Regular Retraining and Updates:** As fraud patterns evolve, the models are regularly retrained with new data to maintain their effectiveness.

Challenges faced

In the **Fraud Detection in Retail Transactions** project, one of the most challenging aspects was dealing with **imbalanced data**, where fraudulent transactions were extremely rare compared to legitimate ones. This imbalance can lead to models that are biased toward predicting legitimate transactions and might miss detecting fraud, which is the key objective of the project.

Challenge: Imbalanced Data

Problem:
Fraudulent transactions were much less frequent than legitimate ones, making the dataset heavily imbalanced (e.g., only 1% of transactions being fraudulent).

In such cases, traditional machine learning models might predict the majority class (legitimate transactions) with high accuracy, but this would lead to poor fraud detection performance (high false negatives). The main challenge was to develop a model that was sensitive enough to identify these rare fraudulent cases without flooding the system with false positives (misclassifying legitimate transactions as fraud).

Impact:
If I didn't address this issue effectively, the fraud detection system might not flag the actual fraudulent transactions in time, leading to financial losses for the retailer. It could also frustrate customers by incorrectly

flagging legitimate transactions, which could harm the retailer's reputation.

Approach to Overcome the Challenge:

Data Resampling (SMOTE and Undersampling):
To combat the data imbalance, I used **Synthetic Minority Over-sampling Technique (SMOTE)**. SMOTE works by generating synthetic examples of fraudulent transactions to balance the dataset. This helps the model learn the characteristics of the minority class (fraudulent transactions) more effectively.

I also used **undersampling** on the majority class (legitimate transactions) to reduce the dominance of the legitimate transactions, ensuring that the model was exposed to more fraud examples during training.

Class Weights Adjustment:
Many algorithms, including **Random Forest** and **XGBoost**, allow you to assign different **weights to classes** to counteract the imbalance. By assigning a higher weight to the fraud class, I ensured that the model paid more attention to fraudulent transactions during training, making it more sensitive to fraud without overfitting.

Model Selection and Hyperparameter Tuning:
I experimented with several models that are known to handle imbalanced data well, such as **Isolation Forest** (an unsupervised anomaly detection model). I tuned the **hyperparameters** of the models, such as the number of trees in Random Forest and the learning rate in XGBoost, to optimize performance and ensure a balanced trade-off between **precision** (reducing false positives) and **recall** (capturing more fraudulent transactions).

Evaluation Metrics Focused on Imbalance:
Instead of using standard accuracy, I focused on evaluation metrics like **precision**, **recall**, and the **F1-score**. These metrics give a clearer picture of how well the model is detecting fraud. I also used **precision-recall curves** to help visualize and select the best decision threshold for the model, ensuring that it didn't flag too many false positives while still detecting as many fraudulent transactions as possible.

Continuous Model Monitoring and Feedback Loop:
After deploying the fraud detection system, I implemented a **feedback loop** to continuously monitor the model's performance in real-time. This allowed me to tweak the model further based on new fraud patterns and ensure it remained effective as fraud tactics evolved. Over time, I refined the model using fresh transaction data, incorporating insights from false positives/negatives into the feature engineering process.

Result:

By addressing the imbalance and optimizing the models, I was able to significantly improve the **detection of fraudulent transactions** while minimizing the number of legitimate transactions flagged as fraudulent. The final system demonstrated high **precision** (minimizing false positives) and **recall** (capturing most fraudulent transactions), which led to a more efficient fraud detection system. This reduced financial losses and improved customer trust in the retailer.

Hello, my name is K. Manish Royal, and I am a Data Scientist with 2 years of experience including 7 months internship experience in the field. My expertise lies in Generative AI, Computer Vision, Machine Learning, and Natural Language Processing (NLP). Over the years, I have worked on various AI models and deployed them to solve practical business problems, utilising a range of advanced algorithms and technologies.

In Generative AI, I have hands-on experience with models like GPT-4 Tarbo (Generative Pre-trained Transformer) focusing on generating creative content such as text, images, and video. I've worked on building chatbots, content generation tools, and other AI-powered solutions customised for specific business needs.

In Computer Vision, I have worked on projects involving object detection, image classification, and face recognition, utilising algorithms such as YOLO (You Only Look Once), ResNet, and OpenCV. I have also implemented custom Convolutional Neural Networks (CNNs) for tasks like image segmentation and video analysis.

For Machine Learning, I am proficient in using algorithms like Random Forest, XGBoost, K-Means Clustering, and Support Vector Machines (SVM). I have developed models for predictive analytics, fraud detection, and customer segmentation, employing techniques such as hyperparameter tuning, cross-validation, and feature engineering to optimise model performance.

In Natural Language Processing (NLP), I have worked on projects like sentiment analysis, text classification, named entity recognition (NER), and language translation. I have utilised models such as BERT (Bidirectional Encoder Representations from Transformers) and LSTM (Long Short-Term Memory networks) to solve a wide range of text-based problems, including chatbot development and text summarisation.

Additionally, I am experienced in using tools like GitHub for version control and Jira for project management and team collaboration.

As a team player, I enjoy working in collaborative environments and sharing knowledge with peers to achieve common goals. I have mentored junior team members, contributed to code debugging, and consistently strive to improve code quality in projects.

I am always keen to learn and apply the latest advancements in AI to deliver impactful solutions. My goal is to leverage data-driven insights and AI to make a positive difference in business outcomes.

Self introduction and projects, skills

most challenging project that you have worked, most in technical terms

Changing of company:

If the interviewer asks, **Actually I have enjoyed my current role and I have gained valuable experience there, and particularly in projects . I want to work more on AI projects so I believe it's the right time to take for me on new challenges that align with my long-term career aspirations.**

About AIML Labs:

AIML Labs Private Limited (ALPL) is a Private Limited Indian Non-Government Company incorporated in India and The Company is engaged in the Education Industry.

About APPLITECH SLOUTIONS:

Applitech Solution is a global IT Consulting & Solutions company that has been providing high value-added business solutions to enterprises of all sizes.

Project explanation based on current project

Starting with AI project and pervious projects

Internship project want to spoke based if the interviewer was asked

Q1: What is data science?

Answer:

Data science is a field that involves using techniques from statistics, mathematics, and computer science to analyze and draw insights from data.

Q2: What skills do I need to be a data scientist?

Answer:

Data scientists typically need skills in statistics, machine learning, data visualization, and programming. Strong communication and critical thinking skills are also important.

Q3: What programming languages should I learn for data science?

Answer:

Some popular programming languages for data science include Python, R, and SQL. It's also helpful to have some familiarity with other languages like Java and C++.

Q4: How long does it take to learn data science?

Answer:

Learning data science is an ongoing process that can take several months to several years, depending on your background and level of experience.

Q5: What kind of jobs can I get with a background in data science?

Answer:

Some common job titles in data science include data analyst, data scientist, machine learning engineer, and business intelligence analyst.