

CS210 Projects

Customer Churn Prediction for a Telecom Company

1. Data Collection:

To collect data, you can use various methods:

- You can use sample data from dummy APIs
- Or you can download a kaggle dataset from having customer reviews

Extract the data in python using pandas/API calls

2. Data Storage:

Choose appropriate databases for different data types:

Relational Database (e.g., PostgreSQL): Use this for structured data like customer profiles, orders, and transactions.

NoSQL Database (e.g., MongoDB): Ideal for unstructured data such as customer reviews, social media posts, or logs.

3. Data Cleaning:

Use Pandas for data cleaning:

Handle missing values (impute or drop them).

Detect and handle outliers.

Address data inconsistencies.

4. Data Transformation:

Feature engineering:

Create new features from existing ones (e.g., aggregations, ratios).

Normalization:

Scale numerical features to a common range (e.g., Min-Max scaling or Z-score normalization).

5. Exploratory Data Analysis (EDA):

Visualize data trends and distributions: (Plot the age groups of the customers)

What does the graphs look like? Is it skewed or normalized

Use libraries like Matplotlib and Seaborn.

6. Perform sentiment analysis on the customer reviews and label them as positive, neutral or negative

7. Add the type of customer review in database and plot the number of reviews given in each type of review

Twitter Api Project

1. Use tweepy API to collect data from twitter. You will need a twitter developer account for it and authorize it with valid credentials. Use proper rate limits on the tweets while collecting data.
2. Setup mongodb connection and store it in mongodb database using pymongo. Convert each tweet to JSON format and insert it into the MongoDB collection.
3. Use Apache Kafka for real-time data streaming. Create a Kafka consumer to read tweets from the Kafka topic and store them in MongoDB.
4. Import re and nltk library to clean tweets like urls(contains https://) and mentions(contains @).
5. Use nltk library to remove english stopwords and remove them from the tweets (words like ‘is’, ‘and’, ‘the’) and then perform tokenization and stemming using nltk library.
6. Create a dictionary of words along with their frequency in the tweets and use matplotlib library to plot words with their frequency.
7. Import and use textblob library in python to perform sentiment analysis and display the number of each positive, negative and neutral tweets and store it in the type of tweet in mongodb(whether it was positive, negative or neutral)
8. Create a real-time dashboard to visualize sentiment analysis results. You can use a streamlit library for this. Fetch data from MongoDB and display text with sentiment.

Healthcare Data Management and Analysis

1. Data Collection

Hospital Databases: Get a kaggle dataset and import the csv file using pandas

2. Data Storage

Objective: Transform the data and store the collected data efficiently, catering to different data structures.

Approach:

Relational Databases (e.g., MySQL): Use for structured data like patient records, demographics, and visit history.

3. Data Cleaning

Objective: Ensure data quality and consistency.

Approach:

Handle Missing Values: Impute or remove missing values as appropriate.

Remove Duplicates: Identify and eliminate duplicate records.

Resolve Inconsistencies: Standardize data formats and correct any inconsistencies in the dataset. (like changing date formats)

4. Data Transformation

Objective: Prepare data for analysis and modeling.

Feature Engineering: Create new features such as categorizing ages into groups or classifying diseases into broader categories.

Normalization and Scaling: Normalize and scale numerical data to ensure it is on a comparable scale for machine learning models.

5. Exploratory Data Analysis

Objective: Understand the data and uncover underlying patterns.

Approach:

Data Visualization Tools: Use tools like Matplotlib, Seaborn, or Tableau to visualize data, identify trends, correlations, and anomalies. Plot the graphs for certain age groups having a disease.

6. Predictive Modeling

Objective: Develop models to predict health-related outcomes.

Approach:

Machine Learning Libraries (e.g., Scikit-learn): Utilize these libraries to build and train models for tasks such as disease outbreak detection and predicting patient readmission rates.

Above given are only examples for predicting variables. You will be predicting variables depending on your own dataset

7. Model Deployment

Objective: Make predictive models accessible and actionable.

Approach:

APIs with Flask/Django: Deploy models as APIs using Plotly or Django

Sales Forecasting for an E-commerce Platform

1. Data Collection: Retrieve sales data from kaggle/uci source which seems suitable for data analysis and time series analysis
2. Data Storage: Store the data in a suitable database of your choice MySQL, AWS or mongodb for unstructured data
3. Data Cleaning: Preprocess the data using Pandas and NumPy. Handle missing values, outliers, and inconsistencies.

Standardize formats (e.g., date formats) and ensure data quality.

4. Data Transformation: Perform feature engineering: Create lag features (e.g., previous day's sales) to capture temporal patterns.

Calculate moving averages or rolling sums to smooth out noise.

Address seasonality and trends:

Decompose time series data to identify seasonal and trend components.

5. Exploratory Data Analysis (EDA):

Visualize sales trends, seasonality, and other patterns using tools like Matplotlib or Seaborn.

Explore correlations between variables and identify potential insights.

6. Forecasting Models: Build time series forecasting models:

ARIMA (AutoRegressive Integrated Moving Average): Suitable for stationary data with autocorrelation and seasonality.

7. Model Evaluation: Metrics: To assess model performance, we use various metrics. Some common ones include: Some common metrics include: Mean squared error for numerical data or cross entropy loss for a classification data

Model Deployment:

Flask/Django: Use Flask or Django to create an API for deploying your model. These frameworks simplify the process and allow your model to handle real-world scenarios efficiently

Building a Recommendation System for an Online Retailer

1. Data Extraction:

Get data from kaggle for a retailer database for which a recommendation system can be developed. Import it using pandas

2. Data Storage

Store data in a Mysql for efficient data analysis:

You may have to normalize the data before storing it

3. Data Cleaning

Clean and preprocess data to ensure quality and consistency.

Data Cleaning with Pandas and NumPy:

Handle missing values (impute or remove).

Standardize data formats.

Remove duplicates if necessary.

4. Data Transformation

Objective:

Transform data for modeling and analysis. (Converting into correct data types) and making new data if necessary.

5. Exploratory Data Analysis (EDA):

Visualize user behavior patterns, product popularity, and correlations.

EDA helps identify insights and potential features for recommendation algorithms.

6. Recommendation Algorithms:

Implement collaborative filtering (user-item interactions), content-based filtering (item features), and hybrid methods (combining both).

Use Python libraries (e.g., Scikit-learn, Surprise) to build and evaluate these algorithms.

7. Model Evaluation and Deployment:

Evaluate recommendation performance using metrics like precision, recall, and F1-score.

Deploy the recommendation system using web frameworks like Flask or Django.

Integrate the system seamlessly with the retailer's website for real-time recommendations.