**CUSTOMER CHURN PREDITCION**

**INNOVATION DESIGN**

Incorporating advanced machine learning techniques, including ensemble models, can significantly enhance prediction accuracy in engineering applications. Ensembles combine the predictions of multiple models to achieve better performance than individual models. Here's how you can integrate ensemble methods into your future engineering predictions:

**1.Understand Ensemble Methods:**

Ensembles work on the principle of diversity among models. Different algorithms or variations in the same algorithm can be combined to reduce overfitting and improve generalization.

Common ensemble methods include Random Forests, Gradient Boosting (e.g., XGBoost, LightGBM), and stacking.

Data Preparation:

Ensure your dataset is well-prepared and pre-processed. This includes handling missing values, scaling features, and encoding categorical variables.

**2.Model Selection:**

Choose diverse base models. For example, if you're using tree-based models, you might choose Random Forest, XG Boost, and Light GBM as your base models. Diversity can come from different algorithms or by varying hyperparameters.

**3.Train Base Models:**

Train each base model on a subset of your data. This subset can be obtained through techniques like bagging (Random Forest) or random sampling.

**4.Combine Predictions:**

Combine predictions from different models. For regression problems, this might involve averaging predictions, while for classification problems, voting or averaging probabilities can be used.

**5.Stacking:**

Stacking involves training a meta-model that takes the predictions of the base models as input. This can be a linear model or another machine learning model. Stacking can capture higher-level patterns in the data.

**6.Hyperparameter Tuning:**

Tune hyperparameters for both base models and the ensemble method. Grid search or random search can be used, depending on the size of your hyperparameter space.

**7.Cross-Validation:**

Use cross-validation to assess the performance of your ensemble. Cross-validation helps to estimate how well the model will generalize to an independent dataset.

**8.Feature Engineering:**

Continue to explore and engineer features that might improve the performance of your models. Feature importance from ensemble models can guide this process.

**9.Regularization:**

Apply regularization techniques to prevent overfitting. This is particularly important when dealing with complex models or stacking

**10.Monitoring and Updating:**

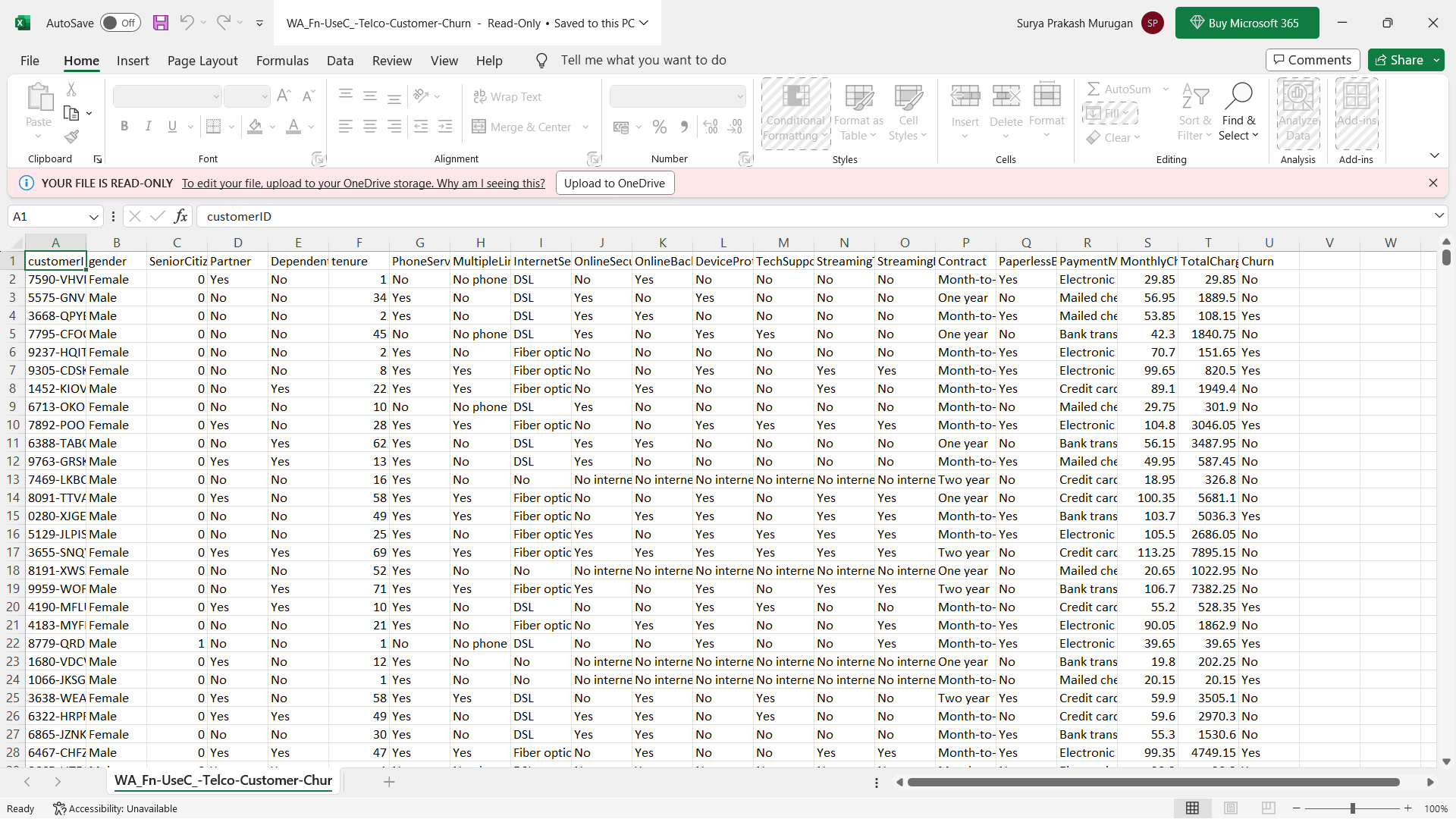
Regularly monitor the performance of your ensemble models and update them as new data becomes available or as the underlying patterns in the data change.

**11.Interpretability:**

While ensembles can improve predictive accuracy, they may be less interpretable than individual models. Consider this trade-off and, if needed, use techniques to interpret ensemble models.

Remember that the effectiveness of ensemble methods can depend on the specific characteristics of your data. Experimentation and thorough evaluation are crucial to finding the best combination of models for your engineering predictions.

Here we are having the dataset:



The given dataset is analysed with the python code and the code is given below:

The code for performing the analysis operation on the customer churn predictions is

**Python code:**

import pandas as pd

import matplotlib.pyplot as plt

# Define the path to the CSV file

csv\_file\_path = "C:\Users\spide\OneDrive\Desktop\WA\_Fn-UseC\_-Telco-Customer-Churn.csv"

try:

# Read the CSV file into a DataFrame

df = pd.read\_csv(csv\_file\_path)

# Assuming your CSV has columns 'Category' and 'Count'

# Change these column names accordingly if needed

category\_column = 'Category'

count\_column = 'Count'

# Create a bar chart

plt.figure(figsize=(10, 6))

plt.bar(df[category\_column], df[count\_column], color='skyblue')

plt.xlabel(category\_column)

plt.ylabel(count\_column)

plt.title('Survey Data Bar Chart')

plt.xticks(rotation=45, ha='right')

# Show the chart

plt.tight\_layout()

plt.show()

except FileNotFoundError:

print(f"File '{csv\_file\_path}' not found.")

except Exception as e:

print(f"An error occurred: {e}")