Telco Churn Prediction Project - Full Journey Documentation

📌Project Overview

Goal: Build and deploy an end-to-end Telco Customer Churn prediction tool using Machine Learning and Streamlit.

Main Components:

- Data Cleaning and Feature Engineering
- Model Training and Evaluation
- Streamlit Dashboard with Visualizations

🧹 Data Cleaning and Preprocessing

- Removed duplicates and handled missing values.
- Converted total charges to numeric and dropped NaNs.
- Encoded binary columns like 'Yes/No' and 'Male/Female'.
- One-hot encoded multi-category columns like Contract, InternetService, etc.

Model Training and Evaluation

🔗Models Tried:

1. **Logistic Regression**
2. **Random Forest**
3. **XGBoost**

🔍Hyperparameter Tuning:

We performed tuning using GridSearchCV with pipelines.

```
param_grid = {
    'classifier__n_estimators': [100, 200],
    'classifier__max_depth': [3, 5, 10]
}
```

📊Best Model Scores and Parameters:

Model: Logistic Regression\ Best Score: 0.8022719204827831\ Best Params: {'classifier__penalty': 'l1', 'classifier__C': np.float64(0.1)}

Model: Random Forest\ Best Score: 0.7983670571529996\ Best Params: {'classifier__n_estimators': 200, 'classifier__min_samples_split': 2, 'classifier__min_samples_leaf': 4, 'classifier__max_depth': 20, 'classifier__bootstrap': False}

Model: XGBoost\ Best Score: 0.8012069577564785\ Best Params: {'classifier__subsample': 0.8, 'classifier__n_estimators': 150, 'classifier__max_depth': 3, 'classifier__learning_rate': 0.1, 'classifier__gamma': 0, 'classifier__colsample_bytree': 1.0}

🔍Limitations of Logistic Regression

While Logistic Regression was a strong baseline, it had some key limitations:

- **Low Recall for Churn class (1):** It correctly identified only 53% of churned customers, which is risky in churn prediction.
- **F1-score imbalance:** The performance on the minority class (churn) was not as strong, leading to an F1-score of just 58%.
- **Linear Assumptions:** It assumes a linear relationship between features and the log-odds of the outcome, which may not capture complex patterns in the data.
- **Sensitivity to Class Imbalance:** Despite using `class_weight='balanced'`, it still struggled to detect many churners accurately.

🔗Final Model:

We used an **Ensemble Voting Classifier** combining the strengths of:

- **Logistic Regression** (interpretable but struggled with class imbalance)
- **Random Forest** (great at handling non-linearity and imbalance)
- **XGBoost** (strong gradient boosting performance)

The ensemble improved performance and reduced individual model drawbacks.

📦Model saved with versioning:

```
joblib.dump(model, f"models/churn_model_{date}.pkl")
```

📊Streamlit app(streamlit_app.py)

The Streamlit web app provides an interactive interface for business users to explore insights and **predict customer churn** based on input features.

🗞1. User Input Interface

- A **sidebar form** allows users to enter customer data (e.g., contract type, monthly charges, internet service, senior citizen, etc.).
- All feature inputs match the format used during training.

🗞️2. Churn Prediction Output

- Once the user clicks **Predict**, the app:
- Runs the prediction on the ensemble model.
- Shows a **Churn Probability** and **Yes/No** prediction.

🔗Output UI Example:

`Prediction: Customer is likely to CHURN. Probability: 74.3%`

📈3. Business Dashboard (Bonus)

In addition to prediction, the app also includes rich visualizations:

- KPI metrics for churned customers
- Churn distribution by contract, tenure, charges, services, demographics
- Actionable segment table to filter risky customers

Sections implemented:

1. Churn Overview (KPI Cards + Donut Chart)
2. Churn by Contract Type (Stacked Bar)
3. Churn by Tenure (Boxplot)
4. Churn by Monthly Charges (Scatter)
5. Churn by Services (Heatmap)
6. Churn by Demographics (Pie Charts)
7. Payment Method & Churn (Treemap)
8. Actionable Segments (Filtered Table)

📌**Error:**

```
ValueError: names='index' not found in DataFrame.
```

🔗**Fix:** Changed `names='index'` to `names=churn_counts.index` in donut chart.

🩲GitHub Setup

🧹GitHub was not recognized in CMD:

```
git : The term 'git' is not recognized
```

🔗**Fix:** Installed Git and added to system PATH.

🔗Git Init + Push:

```
cd telco-churn-app

# Initialize
git init
git add .
git commit -m "Initial commit"

# Connect to GitHub
git remote add origin https://github.com/Amrita-DevX/telco-churn-app
git branch -M main
git push -u origin main
```

Clone the repository and install dependencies:

```
git clone https://github.com/Amrita-DevX/telco-churn-app.git
cd telco-churn-app

# Optional: create virtual environment
python -m venv venv
# Activate it:
# Windows
venv\Scripts\activate
# Mac/Linux
source venv/bin/activate

# Install all dependencies
pip install -r requirements.txt
```

To run the app on your local machine:

```
streamlit run streamlit_app.py
```

Once it starts, open the provided URL in the browser.

🗄️ Deployment Note

Due to compatibility issues on **Streamlit Cloud** and **Render**, we were unable to deploy this app online.

🔩 Common Deployment Errors Faced:

- `ModuleNotFoundError: No module named 'distutils'`
- `numpy==1.26.0` not compatible with Python 3.13
- `pip._vendor.pyproject_hooks._impl.BackendUnavailable`

🕰️Why This Happened:

These platforms currently use **Python 3.13**, while some packages (e.g., NumPy, Pandas, Scikit-learn) are not fully stable or compatible with this version yet.

🔧Advice for Anyone Using This Repo:

If you want to deploy this project on **Render** or **Streamlit Cloud**, we recommend:

- Setting your Python version to `3.10` or `3.9`
- Using `numpy==1.24.3`, `pandas==1.5.3`, and `scikit-learn==1.3.2`
- Including a `runtime.txt` file with this line:

```
python-3.10
```

- Avoiding large files like videos in the repo

This project is 100% working **locally** and produces reliable results and visualizations. Just not yet cloud-deployed due to external dependency conflicts.
