**CUSTOMER CHURN PREDICTION APP: PROJECT REPORT**

**1. Project Overview**

**1.1 Purpose**

The Customer Churn Prediction App is a machine learning-based web application designed to predict whether telecommunications customers are likely to churn (i.e., discontinue their service) based on key features such as tenure, monthly charges, total charges, senior citizen status, and contract type. Built using Python, Streamlit, scikit-learn, and SHAP, the app provides actionable insights for businesses to identify at-risk customers and inform retention strategies. The project also includes an explainability component using SHAP to interpret model predictions, making it valuable for both technical and non-technical stakeholders.

**1.2 Key Features**

* **Data Input Options**: Users can upload a custom CSV file or select from three pre-loaded sample datasets (sample1.csv, sample2.csv, sample3.csv) stored in a sample/ directory.
* **Prediction Output**: The app predicts churn (binary outcome: "Churn" or "No Churn") and provides churn probability as a percentage.
* **Explainability**: SHAP (SHapley Additive exPlanations) generates global (summary) and local (waterfall) plots to explain feature contributions to predictions.
* **Downloadable Results**: Users can download predictions as a CSV file (churn\_predictions.csv) for further analysis.
* **Robust Validation**: Input data is validated for required columns and valid contract values, ensuring reliable predictions.

**1.3 Target Audience**

* **Business Stakeholders**: Customer success or marketing teams in telecommunications who need to identify at-risk customers.
* **Data Scientists/ML Engineers**: Professionals interested in model interpretability and deployment of ML models as web apps.
* **Job Interview Context**: Demonstrates skills in machine learning, data preprocessing, model explainability, and web app development.

**2. Technical Details**

**2.1 Dataset**

The app uses the Telco Customer Churn dataset (assumed to be Telco-Customer-Churn.csv for training). Key features used:

* **tenure**: Number of months the customer has been with the company (numeric).
* **MonthlyCharges**: Monthly bill amount (numeric).
* **TotalCharges**: Total amount billed over the customer’s tenure (numeric, with cleaning for invalid values like spaces).
* **SeniorCitizen**: Whether the customer is a senior citizen (binary: 0 or 1).
* **Contract**: Contract type (categorical: "Month-to-month", "One year", "Two year").
* **Target (Churn)**: Whether the customer churned (binary: "Yes" or "No").

**2.2 Machine Learning Pipeline**

* **Preprocessing**:
  + **Contract Encoding**: Label-encoded Contract as {"Month-to-month": 0, "One year": 1, "Two year": 2} to convert categorical data to numeric.
  + **TotalCharges Cleaning**: Removed rows with invalid TotalCharges (e.g., spaces) and converted to numeric.
  + **Feature Scaling**: Applied StandardScaler to normalize numeric features (tenure, MonthlyCharges, TotalCharges, SeniorCitizen, Contract) for model compatibility.
  + **Target Encoding**: Mapped Churn to {"No": 0, "Yes": 1}.
* **Model**: RandomForestClassifier from scikit-learn, chosen for its robustness to imbalanced data and feature interactions.
  + **Hyperparameter Tuning**: Used GridSearchCV to optimize:
    - n\_estimators: [100, 200]
    - max\_depth: [None, 10, 20]
    - min\_samples\_split: [2, 5]
    - min\_samples\_leaf: [1, 2]
    - Scoring metric: F1-score (to handle class imbalance).
    - Class weight: balanced to address churn’s minority class.
  + **Output**: Trained model saved as model.pkl, scaler as scaler.pkl.
* **Training Script**: A Python script (train\_model.py) automates data loading, preprocessing, model training, and saving.

**2.3 Web App Development**

* **Framework**: Streamlit, a Python library for creating interactive web apps with minimal frontend expertise.
* **Structure**:
  + **Input Section**: Radio buttons allow users to choose between uploading a CSV or selecting a sample dataset from sample/.
  + **Preprocessing**: Mirrors training pipeline (label-encoding, scaling, validation).
  + **Prediction Section**: Displays a table with input data, Prediction (Churn/No Churn), and Probability (churn likelihood as percentage).
  + **Download Feature**: A button to download the results table as churn\_predictions.csv.
  + **SHAP Section**: Visualizes feature importance (summary plot) and individual prediction explanations (waterfall plot).
* **File Management**:
  + Sample datasets stored in sample/ directory, accessed via absolute paths for reliability.
  + Model and scaler loaded from model.pkl and scaler.pkl.

**2.4 Explainability with SHAP**

* **Library**: SHAP (SHapley Additive exPlanations) for model interpretability.
* **Configuration**: TreeExplainer with model\_output="probability" to explain churn probabilities.
* **Outputs**:
  + **Summary Plot**: Shows global feature importance, ranking features by their impact on churn predictions.
  + **Waterfall Plot**: Explains individual predictions by showing how each feature contributes to the final probability.
* **Debugging**: Outputs model predictions vs. SHAP sums to verify additivity (ensuring explanations align with model outputs).

**2.5 Technologies Used**

* **Python Libraries**:
  + scikit-learn (1.6.1): Model training, preprocessing, hyperparameter tuning.
  + pandas: Data manipulation.
  + numpy: Numerical operations.
  + Streamlit: Web app interface.
  + SHAP: Model explainability.
  + joblib: Model and scaler serialization.
  + matplotlib: Plot rendering (via SHAP).
* **Environment**: Managed via a virtual environment (churn-env) with dependencies saved in requirements.txt.
* **OS**: macOS (assumed from /Applications/Ana/anaconda3/envs/churn-env/).

**3. Challenges and Solutions**

**3.1 Challenge: SHAP ExplainerError (Additivity Check Failure)**

* **Issue**: Initial SHAP error: “Additivity check failed in TreeExplainer” (SHAP sum ≈ 0.058, model output ≈ 0.050).
* **Cause**: Mismatch in Contract preprocessing between training (label-encoded) and app (categorical).
* **Solution**:
  + Added label-encoding for Contract in the app to match the training script ({"Month-to-month": 0, "One year": 1, "Two year": 2}).
  + Validated input CSVs for correct Contract values.
  + Outcome: SHAP sums aligned with model outputs, resolving the error.

**3.2 Challenge: 'DecisionTreeClassifier' object has no attribute 'monotonic\_cst'**

* **Issue**: SHAP’s TreeExplainer failed due to missing monotonic\_cst attribute in DecisionTreeClassifier (part of RandomForestClassifier).
* **Cause**: Model trained with an older scikit-learn version (<1.3, where monotonic\_cst was introduced) or environment mismatch.
* **Solution**:
  + Upgraded scikit-learn to 1.6.1 in the churn-env environment.
  + Retrained the model using the updated training script to ensure compatibility.
  + Ensured the same environment was used for training and app execution.
  + Outcome: Error resolved, SHAP plots rendered correctly.

**3.3 Challenge: Sample Datasets Not Loading**

* **Issue**: App failed to load sample CSVs, showing “Sample file not found” errors.
* **Cause**: Directory name mismatch (samples/ in code vs. sample/ in file system).
* **Solution**:
  + Updated app code to use sample/ directory.
  + Used absolute paths (os.path.join(BASE\_DIR, "sample")) to ensure reliable file access.
  + Added debugging output to display sample file paths for verification.
  + Outcome: Sample datasets (sample1.csv, sample2.csv, sample3.csv) loaded successfully.

**3.4 Challenge: Enhancing User Experience**

* **Issue**: Users needed an easier way to test the app and export results.
* **Solution**:
  + Added radio buttons to select from three sample datasets, reducing the need for manual CSV uploads during testing.
  + Implemented a download button to export prediction results as churn\_predictions.csv, placed after the results table for intuitive access.
  + Outcome: Improved usability for both technical and non-technical users.

**4. How to Explain the Project**

**4.1 High-Level Explanation (Non-Technical Audience)**

* **What It Does**: “This app helps telecom companies predict which customers might leave their service, allowing them to take action to keep those customers. Users can upload customer data or use sample datasets, and the app shows who’s likely to churn, why, and lets them download the results.”
* **Why It Matters**: “By identifying at-risk customers early, businesses can save money on retention efforts. The app also explains its predictions in a way that’s easy to understand, building trust in the results.”
* **How It Works**: “It uses a machine learning model trained on customer data like tenure and contract type. The app has a simple interface where you pick a dataset, see predictions, and get visual explanations of what drives those predictions.”

**4.2 Technical Explanation (Job Interview/Data Science Audience)**

* **Problem Statement**: “The goal was to build a churn prediction model for telecom customers and deploy it as an interactive web app with explainable predictions. The dataset included features like tenure, charges, and contract type, with a binary churn target.”
* **Approach**:
  + **Data Preprocessing**: “I cleaned the data by handling invalid TotalCharges values and label-encoded the categorical Contract feature to match the model’s requirements. I used StandardScaler to normalize features for better model performance.”
  + **Model Selection**: “I chose a RandomForestClassifier for its ability to handle imbalanced data and capture feature interactions. I tuned hyperparameters with GridSearchCV using F1-score to optimize for the minority churn class.”
  + **Deployment**: “I built a Streamlit app that allows users to upload CSVs or select sample datasets. The app mirrors the training pipeline for preprocessing and uses SHAP’s TreeExplainer to provide global and local explanations of predictions.”
  + **Challenges**: “I faced a SHAP additivity error due to inconsistent preprocessing, which I fixed by aligning Contract encoding. A monotonic\_cst error required upgrading scikit-learn to 1.6.1 and retraining. I also resolved a file path issue for sample datasets by using absolute paths.”
* **Results**: “The app delivers accurate predictions, interpretable SHAP plots, and a downloadable CSV of results. It’s robust, with input validation and debugging outputs to ensure reliability.”
* **Skills Demonstrated**: “This project showcases my expertise in machine learning (scikit-learn), model interpretability (SHAP), data preprocessing (pandas), web app development (Streamlit), and environment management (virtual environments).”

**4.3 Job Interview Tips**

* **Structure Your Answer** (STAR Method):
  + **Situation**: “I wanted to build a churn prediction tool for telecom customers to demonstrate my ML and deployment skills.”
  + **Task**: “My goal was to create a web app that predicts churn, explains predictions, and supports sample and custom datasets.”
  + **Action**: “I trained a RandomForestClassifier with GridSearchCV, built a Streamlit app, integrated SHAP for explainability, and resolved issues like preprocessing mismatches and scikit-learn compatibility.”
  + **Result**: “The app is user-friendly, provides accurate predictions and explanations, and includes features like downloadable results, making it valuable for both business and technical users.”
* **Highlight Problem-Solving**:
  + Emphasize how you debugged and fixed errors (e.g., ExplainerError, monotonic\_cst, sample directory mismatch).
  + Example: “When I encountered a SHAP error, I traced it to a preprocessing mismatch, aligned the app’s pipeline with the training script, and verified the fix with debugging outputs.”
* **Showcase Impact**:
  + “The app empowers businesses to retain customers by identifying churn risks early, with SHAP explanations ensuring stakeholders trust the model.”
* **Tailor to the Role**:
  + For ML roles: Focus on model selection, tuning, and SHAP.
  + For data engineering: Highlight preprocessing and pipeline consistency.
  + For full-stack or deployment: Emphasize Streamlit and file management.

**5. Key Takeaways for Personal Study**

**5.1 Technical Lessons**

* **Preprocessing Consistency**: Ensuring identical preprocessing (e.g., label-encoding) between training and inference is critical for model and explainer accuracy.
* **Environment Management**: Using a virtual environment and matching library versions (e.g., scikit-learn 1.6.1) prevents compatibility issues.
* **Model Interpretability**: SHAP provides powerful tools for explaining tree-based models, but requires careful configuration (e.g., model\_output="probability").
* **File Path Handling**: Absolute paths (os.path.abspath) are more reliable than relative paths for web apps, especially in Streamlit.
* **Debugging**: Adding outputs (e.g., shapes, feature names, SHAP sums) simplifies troubleshooting.

**5.2 Project Management Lessons**

* **Iterative Debugging**: Breaking down errors (e.g., additivity, monotonic\_cst, file paths) into smaller problems allowed systematic resolution.
* **User-Centric Design**: Features like sample datasets and downloadable results improve usability for diverse audiences.
* **Documentation**: Saving dependencies (requirements.txt) and maintaining clear code comments ensure reproducibility.

**5.3 Skills Gained**

* **Machine Learning**: Model training, hyperparameter tuning, handling imbalanced data.
* **Data Preprocessing**: Cleaning, encoding, scaling with pandas and scikit-learn.
* **Explainability**: Implementing SHAP for global and local interpretability.
* **Web Development**: Building interactive apps with Streamlit.
* **Problem-Solving**: Diagnosing and resolving complex errors through debugging and research.
* **File Management**: Organizing datasets and ensuring robust path handling.

**6. How to Present in a Job Interview**

**6.1 Slide Deck Outline (Optional)**

If preparing a presentation:

1. **Introduction**: Project goal and business value (churn prediction for telecom).
2. **Data and Model**: Dataset features, RandomForestClassifier, preprocessing pipeline.
3. **App Features**: Streamlit interface, sample datasets, predictions, SHAP plots, download button.
4. **Challenges and Solutions**: Key issues (additivity, monotonic\_cst, sample paths) and how you resolved them.
5. **Demo**: Screenshots or live demo of the app (if possible).
6. **Impact and Future Work**: Business benefits and potential enhancements (e.g., real-time API, more features).
7. **Q&A**: Be ready to discuss technical details or trade-offs.

**6.2 Common Interview Questions and Answers**

* **Q: Why did you choose RandomForestClassifier?**
  + A: “It’s robust for imbalanced datasets like churn, captures non-linear relationships, and works well with mixed feature types. I tuned it with GridSearchCV to optimize performance.”
* **Q: How does SHAP help in this project?**
  + A: “SHAP explains why the model predicts churn by showing feature contributions. The summary plot highlights key drivers like Contract, while the waterfall plot details individual predictions, building trust for stakeholders.”
* **Q: How did you handle the additivity error?**
  + A: “I found a preprocessing mismatch in Contract encoding. I aligned the app’s pipeline with the training script by label-encoding Contract, validated inputs, and verified SHAP sums matched model outputs.”
* **Q: What was the hardest part?**
  + A: “The monotonic\_cst error was tricky because it stemmed from a scikit-learn version mismatch. I upgraded to 1.6.1, retrained the model, and ensured environment consistency, which taught me the importance of version control.”
* **Q: How would you improve the app?**
  + A: “I’d add real-time data integration via an API, support more features (e.g., payment method), and enhance the UI with interactive filters for results.”

**6.3 Visual Aids**

* **Screenshots**: Show the app’s interface (data selection, results table, SHAP plots).
* **SHAP Plots**: Highlight a summary plot to explain feature importance (e.g., “Contract has the highest impact on churn”).
* **Code Snippets**: Share key sections (e.g., preprocessing, SHAP setup) in interviews requiring code discussion.
* **Flow Diagram**: Sketch the pipeline: Data Input → Preprocessing → Model Prediction → SHAP Explanation → Output.

**7. Future Improvements**

* **Additional Features**: Incorporate more dataset features (e.g., payment method, internet service) to improve model accuracy.
* **Real-Time Predictions**: Integrate an API for live data feeds.
* **UI Enhancements**: Add filters for results (e.g., show only churned customers) or interactive SHAP visualizations.
* **Model Alternatives**: Experiment with gradient boosting (e.g., XGBoost) or neural networks for comparison.
* **Deployment**: Host the app on a cloud platform (e.g., Streamlit Community Cloud) for broader access.

**8. Conclusion**

The Customer Churn Prediction App is a comprehensive project that demonstrates end-to-end machine learning expertise, from data preprocessing and model training to deployment and interpretability. By overcoming challenges like SHAP errors, scikit-learn compatibility, and file path issues, I developed a robust, user-friendly tool that delivers valuable insights for telecom businesses. This project highlights my ability to translate complex ML models into practical applications, making it a strong portfolio piece for job interviews or discussions with stakeholders.

**Author**: [Your Name]  
**Date**: June 15, 2025  
**Contact**: [Your Email or LinkedIn] (optional for personal use)  
**Repository**: [Link to GitHub repo if applicable] (optional)

https://drive.google.com/file/d/1xdli0rxCe6K3DkMqRv9sB1yqiBAQCDGj/view?

"https://drive.google.com/uc?export=download&id=1xdli0rxCe6K3DkMqRv9sB1yqiBAQCDGj"