# **Phase 3: Model Training & Validation**

## This phase was done by :Malak Alaa Mohamed

**1. Problem Definition**

The task is to **predict customer churn** (whether a customer will leave the telecom service) based on customer demographics, service usage, and billing information.  
This is a **binary classification problem**:

* **Churn = 1** → customer will leave.
* **Churn = 0** → customer stays.

**2. Data Preparation**

* **Target Variable (Churn)**: Converted to binary numeric values (Yes → 1, No → 0).
* **Features (X)**: All other customer attributes (numeric + categorical).
* **Train-Test Split**: 80% training, 20% testing, with stratification to preserve churn distribution.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42, stratify=y

)

**3. Feature Engineering & Preprocessing**

* **Numeric Features**: Scaled using StandardScaler.
* **Categorical Features**: Encoded using OneHotEncoder (ignore unknown categories in test).
* **ColumnTransformer**: Applied the preprocessing pipeline automatically to the right feature types.

preprocessor = ColumnTransformer(

transformers=[

("num", StandardScaler(), numeric\_features),

("cat", OneHotEncoder(handle\_unknown="ignore"), categorical\_features)

]

)

**4. Models Trained**

Four models were benchmarked:

1. **Logistic Regression** → Linear, interpretable, good baseline.
2. **Random Forest** → Bagging-based ensemble of decision trees.
3. **Support Vector Machine (SVM)** → Kernel-based classifier with probability calibration.
4. **XGBoost** → Gradient boosting with regularization, powerful for tabular data.

Each model was wrapped in a Pipeline with preprocessing included:

pipe = Pipeline(steps=[("preprocessor", preprocessor),

("classifier", model)])

**5. Model Selection**

* Models were trained on the training set and evaluated on the test set using **accuracy** as the first metric.
* The best model was chosen based on test performance.

**Performance Results:**

Logistic Regression: 0.800

Random Forest: 0.773

SVM: 0.796

XGBoost: 0.784

**Best model**: Logistic Regression (accuracy = 80%)

**6. Evaluation Metrics**

Since churn datasets are typically **imbalanced**, multiple metrics were used:

1. **Confusion Matrix**  
   Shows the breakdown of true/false positives/negatives.
2. **Precision-Recall Curve (PRC)**  
   Especially useful in imbalanced classification.
3. **ROC Curve & AUC**  
   Evaluates model’s ability to discriminate between churners and non-churners.
4. **Classification Report**
   * Precision → How many predicted churners were actually churners.
   * Recall → How many actual churners were correctly identified.
   * F1-Score → Balance of precision and recall.
   * Accuracy → Overall correct predictions.

**Final Report (Logistic Regression):**

Accuracy: 80.0%

Precision (Churn=1): 0.653

Recall (Churn=1): 0.522

F1-score (Churn=1): 0.580

ROC-AUC Score: 0.842

**7. Model Persistence**

The best-performing pipeline (Logistic Regression + preprocessing) was saved for later inference:

joblib.dump(best\_model, "best\_churn\_model.pkl")

**8. Inference Function**

A helper function was implemented to predict churn for **a single new customer**.  
It loads the saved model, preprocesses the input, and outputs both class and probability.

def predict\_churn(single\_row: pd.DataFrame):

"""

Predict customer churn for a single row of input.

Args:

single\_row (pd.DataFrame): One row of customer data (same features as training set).

Returns:

dict: {"Churn Prediction": 0/1, "Churn Probability": float}

"""

model = joblib.load("best\_churn\_model.pkl")

pred = model.predict(single\_row)[0]

prob = model.predict\_proba(single\_row)[0,1]

return {"Churn Prediction": pred, "Churn Probability": prob}

**Example Usage:**

new\_customer = X\_test.iloc[[0]]

print(predict\_churn(new\_customer))

Output:

{'Churn Prediction': 1, 'Churn Probability': 0.695}

**9. Key Takeaways**

* **Logistic Regression** outperformed more complex models in this dataset (likely due to feature scaling and linear separability).
* **ROC-AUC (0.842)** indicates strong discriminative power.
* Precision-Recall balance shows the model is fairly good at catching churners, though recall could be improved with resampling or class weights.
* The pipeline approach ensures **scalable deployment** and **reproducibility**.

A diagram of a customer churn prediction

AI-generated content may be incorrect.