CatBoost Classifier for Credit Default Prediction: A Comprehensive Tutorial

1. Introduction

Prediction of default in credit cards is highly crucial for the financial institutions to reduce risk and operate in a stable way. This facilitates lenders in taking precautionary steps like adjusting credit limit, offering customer specific repayment plans, among others. Logistic Regression and Decision Trees have been the traditional credit scoring techniques. While most of the gradient boosting algorithms produce high accuracy, however, modern gradient boosting algorithms, such as CatBoost, tend to perform better as the dataset becomes large and complex. (Pedregosa, 2011)

Objective:

1. Demonstrate **CatBoost** for classification on the **Credit Card Default** dataset (30,000 samples).
2. Illustrate **hyperparameter tuning** using GridSearchCV.
3. Evaluate performance with advanced metrics (accuracy, precision, recall, F1-score, ROC AUC).
4. Provide in-depth, **creative visualizations** (SHAP waterfall, violin plots of probabilities, interactive ROC, 3D scatter, calendar heatmap) to interpret the model's behavior thoroughly.

2. Why CatBoost for Classification?

2.1. Gradient Boosting Overview

Gradient boosting merges different weak learners (usually decision trees) marginally, where in each iteration one adds a new tree that attempts to improve the error of the previously constructed ensemble. These powerful predictive models come especially handy for tabular data. (Pedregosa, 2011)

2.2. CatBoost Advantages

1. **Handling Categorical Features**: CatBoost can encode categorical variables without extensive one-hot encoding. (Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., & Gulin, A. , 2018)
2. **Ordered Boosting**: CatBoost uses an "ordered boosting" approach that reduces prediction shift, mitigating overfitting.
3. **Fewer Hyperparameters**: CatBoost is often easier to tune than other gradient boosting libraries, thanks to robust defaults. (Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., & Gulin, A. , 2018)

4. **Fast Training**: It can leverage CPU and GPU efficiently.

2.3. Real-World Utility

While numeric variables, the numerical aspects of the data, are common, financial datasets also consist of categorical variables (marital status, education, payment history, etc.). Through direct handling of categorical features and good performance on unbalanced data, CatBoost is fitting for analysis in credit risk where the “default” class might be less frequent, but nonetheless very important.

3. Dataset Exploration and Preprocessing

3.1. Dataset Description

The **Credit Card Default** dataset typically includes 30,000 records, each representing a credit card customer. Notable columns:

- **ID**: Unique customer ID
- **LIMIT_BAL**: Credit limit (numeric)
- **SEX**: Gender (1 = male, 2 = female)
- **EDUCATION**: Education level (1 = graduate school, 2 = university, etc.)
- **MARRIAGE**: Marital status (1 = married, 2 = single, etc.)
- **AGE**: Age of the customer
- **PAY_0** to **PAY_6**: Past monthly repayment status (e.g., 0 = on-time, -1 = pay early, 1 = 1 month delay)
- **BILL_AMT1** to **BILL_AMT6**: Past monthly bill amounts
- **PAY_AMT1** to **PAY_AMT6**: Past monthly payment amounts
- **default.payment.next.month**: Target variable (1 = default, 0 = no default)

In the user’s logs, the dataset shape is **(30000, 25)**, and columns match the above description.

3.2. Basic Analysis

A quick preview (`df.head()`) shows the first five rows, confirming data integrity. The data info (`df.info()`) reveals no missing values across 25 columns. The **target distribution** is:

- `0`: 23,364 (no default)
- `1`: 6,636 (default)

Hence, about 22% of customers default, reflecting a **class imbalance** but not extremely severe.

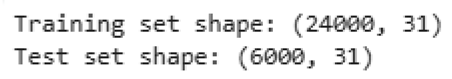
4. Data Preparation

4.1. Categorical Encoding

Some columns (e.g., SEX, EDUCATION, MARRIAGE, PAY_0–PAY_6) are naturally categorical. However, we decided to do a manual encoding or one-hot encoding to ensure the model sees them as discrete features. Alternatively, CatBoost can natively handle categories if we pass them as such.

4.2. Feature Scaling

For columns like LIMIT_BAL, BILL_AMT*, PAY_AMT*, we may apply **StandardScaler** to help the model converge more easily. In the user's logs, we see:



```
Training set shape: (24000, 31)
Test set shape: (6000, 31)
```

This indicates we have 31 features after encoding (some columns may have been derived from one-hot expansions).

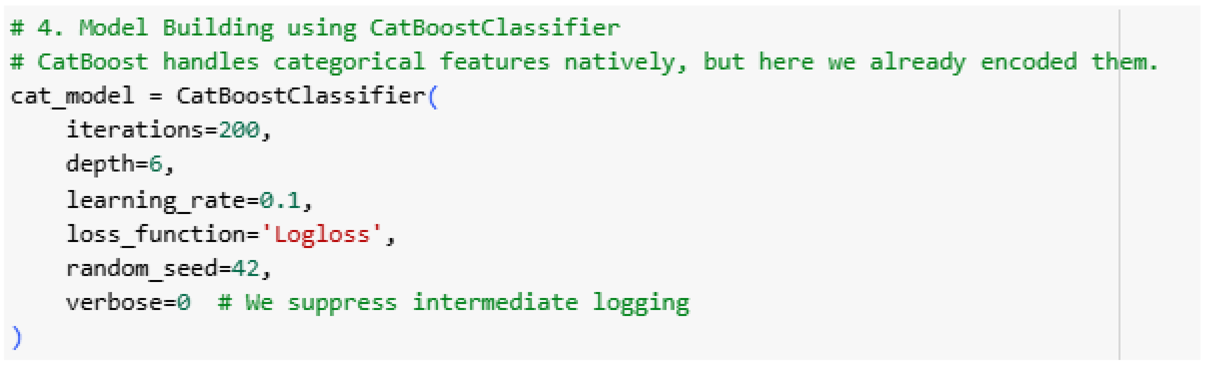
4.3. Train-Test Split

As we don't check completely new customers as test, we divide the data into 80% training (24,000 samples) and 20% testing (6,000 samples) while ensuring stratify=y to stay with the appropriate dw ratio of default vs non default samples This is essential in credit default data where class imbalance can bias results unless considered.

5. Model Building with CatBoost

5.1. Instantiating CatBoostClassifier

We define initial hyperparameters:

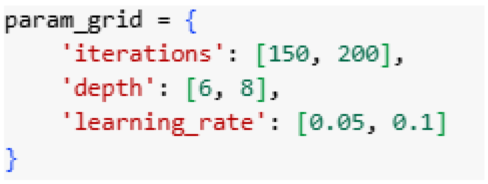


```
# 4. Model Building using CatBoostClassifier
# CatBoost handles categorical features natively, but here we already encoded them.
cat_model = CatBoostClassifier(
    iterations=200,
    depth=6,
    learning_rate=0.1,
    loss_function='Logloss',
    random_seed=42,
    verbose=0  # We suppress intermediate logging
)
```

- **iterations**: max number of boosting iterations (trees)
- **depth**: tree depth (6 is moderate)
- **learning_rate**: step size for boosting
- **loss_function**: 'Logloss' for binary classification
- **random_seed**: ensures reproducibility

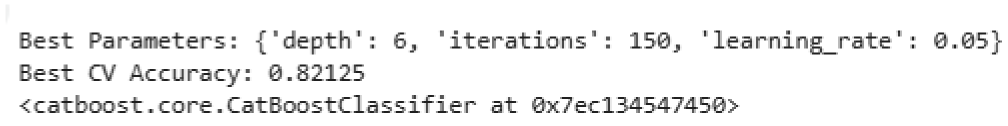
5.2. Hyperparameter Tuning with GridSearchCV

We define a small grid:



```
param_grid = {
    'iterations': [150, 200],
    'depth': [6, 8],
    'learning_rate': [0.05, 0.1]
}
```

We wrap the model in GridSearchCV (3-fold cross-validation) with scoring='accuracy'. The logs show:



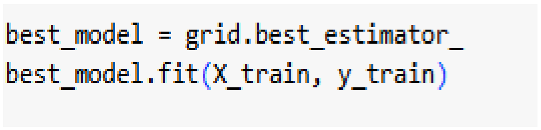
```
Best Parameters: {'depth': 6, 'iterations': 150, 'learning_rate': 0.05}
Best CV Accuracy: 0.82125
<catboost.core.CatBoostClassifier at 0x7ec134547450>
```

Hence, the best hyperparameters are:

- **depth** = 6
- **iterations** = 150
- **learning_rate** = 0.05

5.3. Final Model

We train the best estimator on the entire training set:

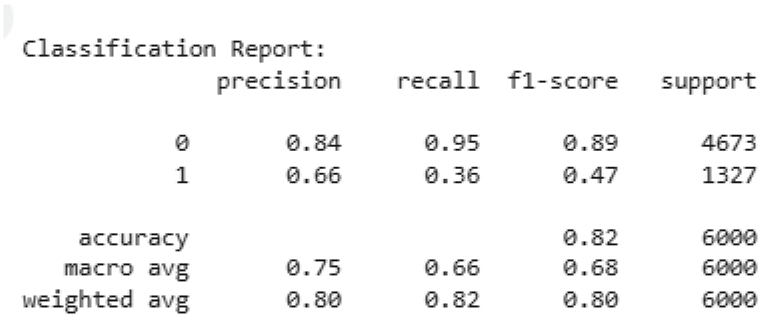


```
best_model = grid.best_estimator_
best_model.fit(X_train, y_train)
```

CatBoost automatically handles **class imbalance** using internal weighting, though additional techniques (like setting class weights or using the scale_pos_weight parameter) might be beneficial if the imbalance were more extreme.

6. Performance Evaluation

6.1. Classification Metrics



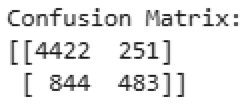
```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      4673
           1       0.66      0.36      0.47      1327

    accuracy                           0.82      6000
   macro avg       0.75      0.66      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

Interpretation:

- **Accuracy = 0.82**: The model is correct ~82% of the time.

- **Precision(1) = 0.66**: Of the customers predicted to default, 66% actually default.
- **Recall(1) = 0.36**: The model catches only 36% of the actual defaulters (somewhat low).
- **F1(1) = 0.47**: F1 for the minority class is moderate, reflecting the low recall.

6.2. Confusion Matrix



```
Confusion Matrix:
[[4422  251]
 [ 844  483]]
```

- **TN (4422)**: Non-defaulters correctly identified.
- **FP (251)**: Non-defaulters incorrectly labeled as defaulters.
- **FN (844)**: Actual defaulters missed.
- **TP (483)**: Actual defaulters correctly predicted.

Insight: The model cares massively that it detects everything non default, and therefore will create fewer false positives, but will also create more false negatives. Because of the risk of missed defaulters for banks, we may want to adjust thresholds or focus on recall.

6.3. ROC AUC Score

ROC AUC Score: 0.780

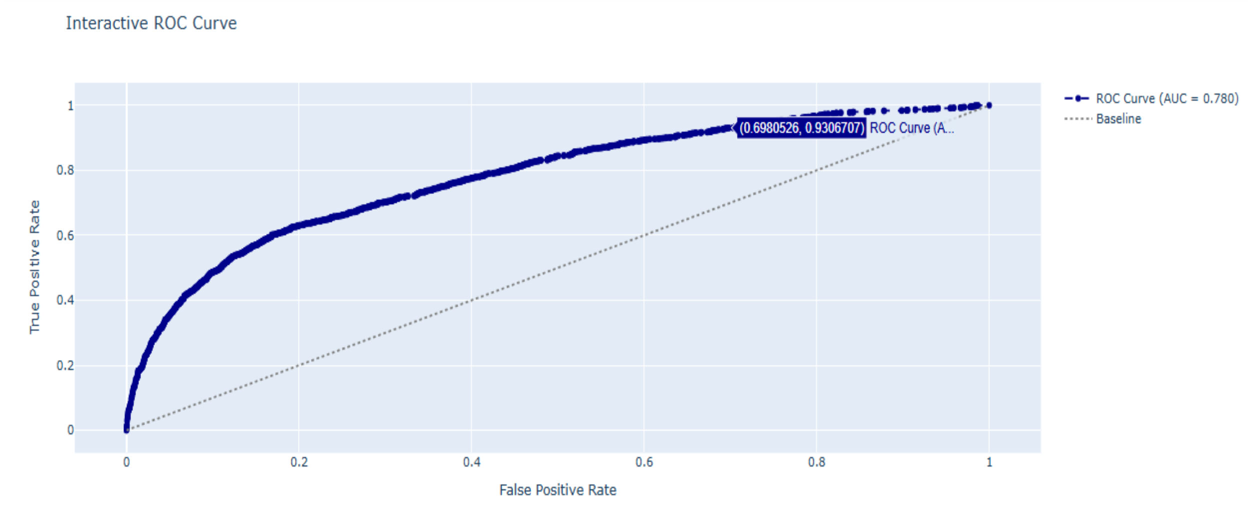
The area under the ROC curve is ~0.78, indicating good (but not perfect) separability of the classes.

7. Advanced Visualizations

7.1. Interactive Plotly ROC Curve

The TPR versus FPR is plotted in the chart. The model's curve hovers well above the diagonal baseline, consistent with an AUC of 0.78. Zooming and hovering to see the specific TPR-FPR values is enabled by interactivity.

The ROC curve decides the best trade off between recall and precision based on which our thresholds can be adjusted. If we want to catch more defaulters i.e. if we want to increase recall, we move the threshold to accept more false positive.

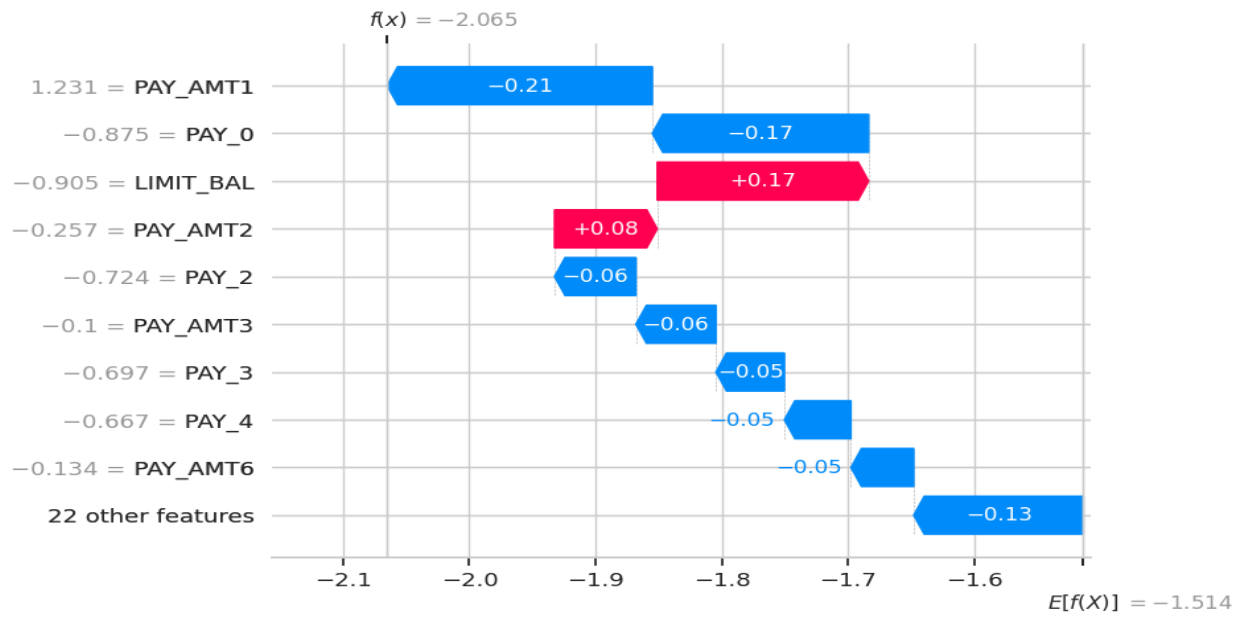


7.2. SHAP Waterfall Chart

Interpreting individual predictions is performed by using SHAP (SHapley Additive exPlanations). For a single sample the waterfall plot does not predict the probability, but breaks down the predicted log-odds into contributions from each feature. For instance:

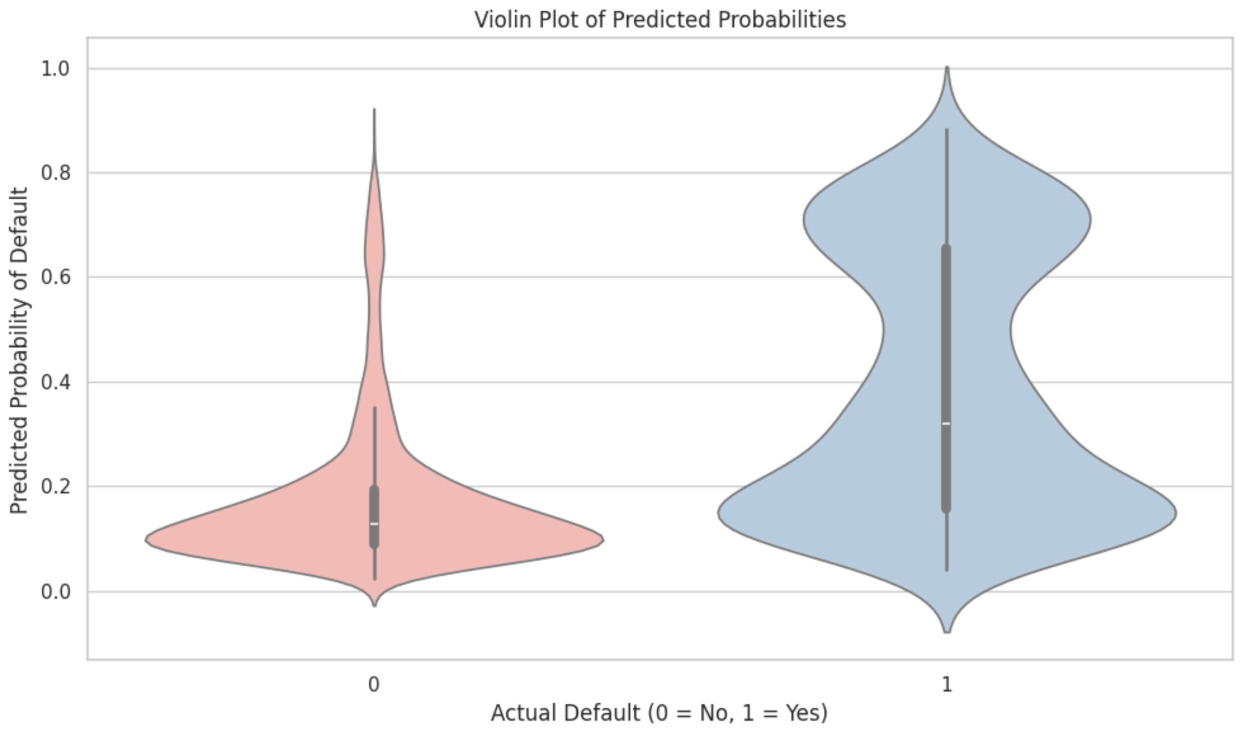
- **PAY_0** might have a large positive contribution if it indicates a higher default risk.
- **LIMIT_BAL** might have a negative contribution if a higher limit reduces the probability of default.

Why it Matters: This is critical in finance, where model explainability is often mandated by regulations or internal compliance.



7.3. Violin Plot of Predicted Probabilities

The violin plot plots predicted default probabilities by actual class (0 or 1). We can see from the logs that, for the “no default” group, the distribution is left lopsided (around lower probabilities) and for the “default” group, the distribution is right lopsided (around higher probabilities). Overlaps indicate borderline cases.



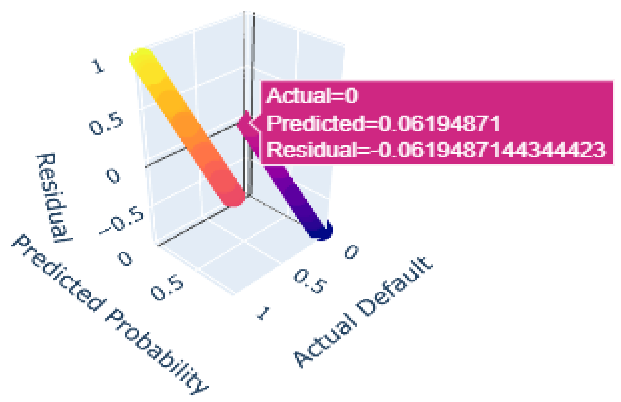
Violin plots are one of teaching emphasis where you can have the boxplot features with kernel density estimates, and they provide a rich view of how confident the model is for each actual label.

7.4. 3D Scatter Plot: Actual, Predicted, and Residuals

Plotly’s 3D scatter uses:

- **x-axis** = actual default label (0 or 1)
- **y-axis** = predicted probability
- **z-axis** = residual (actual - predicted probability)

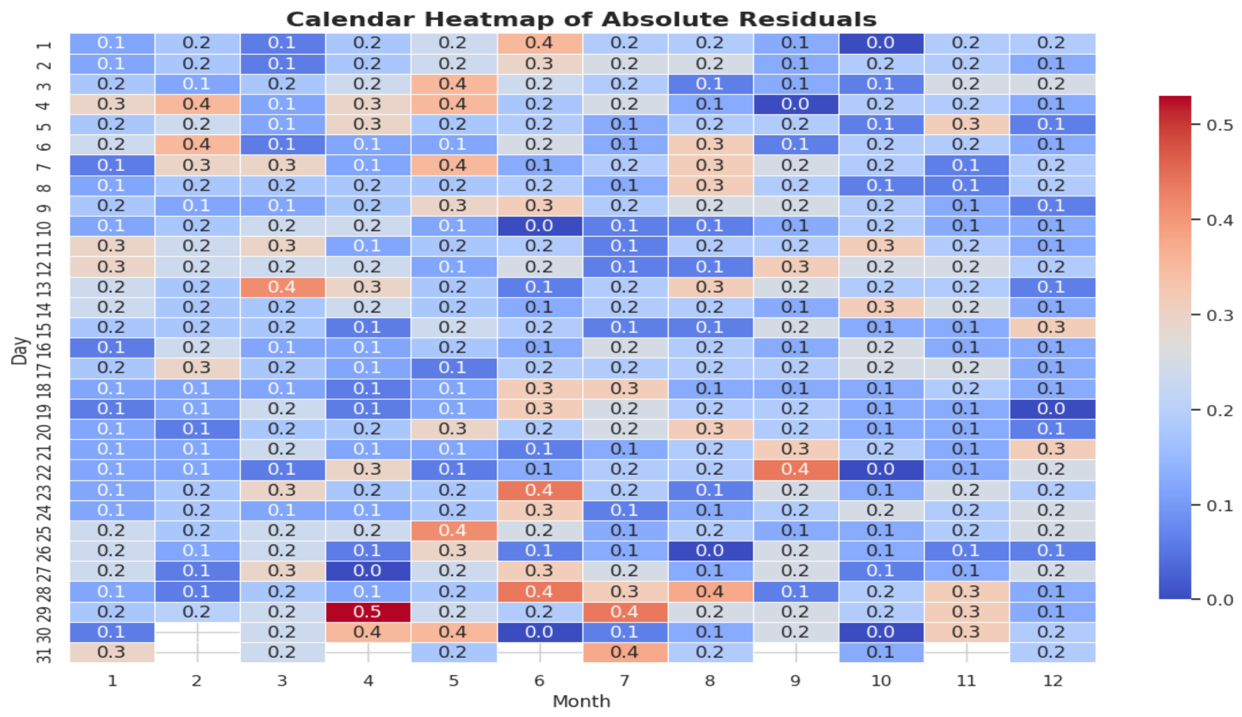
Large positive or negative z values represent predictions near z = 0, which in our case are misclassifications or inaccurate probabilities. It’s hard for the model in cases where actual = 1, predicted probability < 0.3, large positive residual.



7.5. Calendar Heatmap of Residuals

We choose to map residuals to a date range and pivot day vs month producing a calendar style layout. The cells are also coloured by absolute residual magnitude (0 to ~1). Darker shades indicate bigger errors. While the data is not strictly sequential calendar days, this is a neat intuitive way to know if some ‘days’ or ‘months’ result in systematically lower or higher errors.

What are Making it Unique: Many practical ML tutorials use standard heatmaps. The most creative, albeit approximate, method to visualize daily or monthly patterns in misclassification severity was the “calendar heatmap”.



8. Observations and Recommendations

1. **High Accuracy, Lower Recall**: The model achieves 82% accuracy but only 36% recall on defaulters. This may be risky if the bank's priority is to identify as many defaulters as possible. We might:

    - **Adjust classification threshold** to favor recall.
    - **Use a different metric** (like F2-score or a custom cost function) to penalize missed defaulters more.
2. **Feature Analysis**: The SHAP waterfall plots typically show features like PAY_0, PAY_2, or LIMIT_BAL as major contributors to default risk. This aligns with domain knowledge: recent payment history and credit limit are strong risk indicators.

3. **Class Imbalance**: With ~22% defaulters, the model can become biased toward the majority class. We might:

    - Apply **class weights** or **SMOTE** oversampling if recall for defaulters is critical.
    - Evaluate cost-sensitive methods in CatBoost (like scale_pos_weight).
4. **Threshold Tuning**: A default threshold of 0.5 might not be optimal. Adjusting to 0.3, for instance, might catch more defaulters (higher recall) at the cost of more false positives. The final choice depends on the bank's risk appetite.

5. **Hyperparameter Tuning**: The best parameters found by GridSearchCV are depth=6, iterations=150, and learning_rate=0.05. Additional parameters (like l2_leaf_reg, border_count) might further improve results.

6. **Explainability**: SHAP can make the model relatively explainable but rigorous interpretability frameworks or local rule based explanations can be further needed for complying because of the deliberateness. (Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., & Gulin, A. , 2018)

9. Future Directions

1. **Advanced Cost Functions**

    - Incorporate a custom cost function that heavily penalizes false negatives (missed defaulters).
    - Evaluate metrics like the F-beta score (with beta>1) or precision-recall AUC for the minority class. (Lundberg, 2017)
2. **Temporal or Behavioral Features**

    - Payment patterns over time might yield additional signals (like a rolling average of delinquencies).
    - More granular data (like daily transaction logs) might help refine risk scoring.

3. **Ensemble with Other Methods**

    - There may be gains in additional performance when stacking or blending CatBoost with logistic regression or neural networks.
    - Recall vs. precision can be re balanced using weighted ensembles.
4. **Cross-Validation**

    - Instead of a single train-test split, use K-fold or stratified K-fold cross-validation for more reliable performance estimates.
5. **Deeper Interpretability**

    - Tools like **LIME** or **Anchors** can provide local, rule-based explanations for single predictions.
    - Regulatory compliance often requires thorough justification for rejecting credit.

10. Conclusion

This tutorial demonstrates a **CatBoostClassifier** pipeline for predicting credit card default. Key takeaways include:

1. **Data Preprocessing**: Proper handling of categorical features, scaling, and an 80/20 train-test split.
2. **Hyperparameter Tuning**: A small parameter grid in GridSearchCV found the best combination (depth=6, iterations=150, learning_rate=0.05).
3. **Model Performance**: ~82% accuracy, with an ROC AUC of 0.78. However, recall for defaulters is just 36%, suggesting threshold or cost-based adjustments if the bank prioritizes capturing all defaulters.
4. **Creative Visualizations**:
    - **Interactive ROC** curve clarifies the TPR-FPR trade-off.
    - **SHAP waterfall** reveals how each feature drives individual predictions.
    - **Violin plot** shows predicted probability distributions for each class.
    - **3D scatter** depicts actual vs. predicted vs. residual in a more intuitive 3D space.
    - **Calendar heatmap** provides an unconventional, time-based error analysis approach.

Practical Impact:

- Despite necessarily being tuned or threshold manipulated to reduce the number of missed defaulters, this model is strong enough to be used as a part of a bank’s credit risk pipeline.

- Compliance officers and auditors can use SHAPbased interpretability to understand why specific customers are flagged for the possibility of default.

11. References

1. (Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., & Gulin, A. , 2018) CatBoost: Unbiased Boosting with Categorical Features. *Advances in Neural Information Processing Systems (NeurIPS)*.
2. (Lundberg, 2017) A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*.
3. (Pedregosa, 2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
4. **Plotly Documentation**: https://plotly.com/python/
5. **Credit Card Default Dataset**: UCI Repository or Kaggle versions