CREDIT RISK ASSESSMENT AND STRATEGY DESIGN WITH MACHINE LEARNING

Slide 1: Executive Summary

- **Project Goal**: Develop a credit risk model to predict default probabilities and design effective credit assignment strategies.
- **Strategy Evaluation**: We evaluated multiple thresholds to define two approaches:
  - *Conservative Strategy:* Focuses on minimizing risk by approving fewer customers using stricter default thresholds.
  - *Aggressive Strategy*: Focuses on maximizing revenue by approving more customers with more lenient thresholds..
- **Findings**: Our recommended strategies balance customer acceptance and revenue, ensuring the default rate stays below ***10%.***

	Train		
	# Total	Default Rate	Revenue (unit)
Co. Str	44,147	3.51%	$ 397.50
Ag. Str	48,536	6.92%	$ 578.90

	Test 1		
	Total	Default Rate	Revenue (unit)
Co. Str	8,939	3.11%	$ 13.50
Ag. Str	10,395	7.98%	$ 28.00

	Test 2		
	# Total	Default Rate	Revenue (unit)
Co. Str	9,425	4.56%	$ 18.60
Ag. Str	10,446	8.36%	$ 30.20

Slide 2: Data Description

Data Overview

- Data from November 2017 to April 2018.
- **Features:** The dataset contains information on balances, spending patterns, and repayment behavior.
- **Default Definition**: In this project, **default** refers to customers who missed payments for 90 days or more within 12 months of opening their accounts.
  - **1**: Customer defaulted within 12 months.
  - **0**: Customer did not default.
- **Recent Data:** Using the more recent April 2018 period helps predict default risk within the first few months of account opening.

# Months	# Observations	Default_Rate
All Applications	1107069	24.65%
1 Months	984	33.13%
2 Months	2434	31.22%
3 Months	3474	35.75%
4 Months	3752	43.07%
5 Months	4665	39.44%
6 Months	6654	41.30%
7 Months	7322	41.49%
8 Months	9352	45.08%
9 Months	11502	43.58%
10 Months	13290	46.58%
11 Months	12749	44.09%
12 Months	25380	37.87%
13 Months	1005511	22.94%

Slide 3: Features

- **Balance Features (B_):** Capture how much credit customers are utilizing.
- **Spending Features (S_):** High spending without repayment poses credit risks.
- **Duration Features (D_):** Track delays in payments to identify delinquency patterns early, serving as key indicators of future default risk.
- **Payment & Risk Features (P_ & R_):** Monitor consistent payments or debt accumulation and highlight high-risk customers requiring closer monitoring.

Category	# of Features	Percentage
P_	3	1.59%
B_	40	21.16%
D_	96	50.79%
R_	28	14.81%
S_	21	11.11%
Total	189	100.00%

Slide 4: Features Engineering

Feature	Min	1 Percentile	5 Percentile	Median	95 Percentile	99 Percentile	Max	Mean	% Missing
P_2	-0.08062	-0.080577	0.11723767	0.6796678	0.976983645	1.006107229	1.006108	0.63257806	0
S_3_Ave_6	0	0	0	0.1631878	0.535043899	0.863542025	0.86378	0.19864586	0
D_48	0	0	0	0.234495	0.952402771	1.041153288	1.041172	0.36386397	0
D_42_Ave_6	0	0	0	0	0.279871486	0.631599053	0.63162	0.04121258	0
D_50	0	0	0	0	0.309118768	0.692991763	0.693187	0.0704354	0

Slide #5. Data Processing / One-Hot Encoding

GB	GC	GD	GE	GF	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS	GT
D_140	D_141	D_142	D_143	D_144	D_145	D_63_CL	D_63_CO	D_63_CR	D_63_XL	D_63_XM	D_63_XZ	D_63_nan	D_64_-1	D_64_O	D_64_R	D_64_U	D_64_nan	
.001566	0.006516		0.00619	0.00176	0.008869	0	0	1	0	0	0	0	0	0	0	1	0	
.002355	0.00439		0.005739	0.005449	0.001821	0	1	0	0	0	0	0	0	0	1	0	0	
.000127	0.006551		0.003141	0.001077	0.009509	0	1	0	0	0	0	0	0	0	1	0	0	
0.00546	0.005014		0.005464	0.006205	0.002884	0	1	0	0	0	0	0	0	1	0	0	0	
.005929	0.002895		0.000584	0.001526	0.009778	0	0	1	0	0	0	0	0	1	0	0	0	
.001855	0.006451		0.002112	0.00162	0.007749	0	1	0	0	0	0	0	0	0	0	1	0	
.002766	0.009576		0.009199	0.00467	0.009764	0	1	0	0	0	0	0	0	1	0	0	0	
.009429	0.003099		0.009067	0.008524	0.00629	0	0	1	0	0	0	0	0	1	0	0	0	
.003135	0.009202		0.00622	0.005421	0.000262	0	1	0	0	0	0	0	0	1	0	0	0	
.009986	0.008052		0.003689	0.001386	0.001899	0	1	0	0	0	0	0	0	0	1	0	0	
1773624	0.004421		0.00308	0.009005	0.002734	0	1	0	0	0	0	0	0	0	0	1	0	
0.004	0.009894		0.009928	0.000309	0.002175	1	0	0	0	0	0	0	0	0	1	0	0	

Slide #6. Feature Selection

- **Objective**: Feature selection helps enhance model accuracy and efficiency *by filtering out* irrelevant or redundant variables, ensuring we focus only on impactful features.
- **XGBoost for Feature Ranking**: We used XGBoost models to evaluate and rank features based on their importance scores, eliminating those with low predictive value.
- **Outcome**: The selected features from different categories were used in the *grid search process* to fine-tune the model's performance.
- The graph and table on the slide illustrate the **feature selection process** and **number of features selected** from each category. An attached Excel file contains detailed feature importance results for both models

Feature_Default	Feature_Importance
P_2	0.22667983
D_39	0.006678816
B_1	0.11330252
B_2	0.02003743
R_1	0.011736438
S_3	0.003994869
D_41	0.007519789
B_3	0.006313301
D_42	0.005316802
D_43	0.002152707
D_44	0.007129638
B_4	0.004428933
D_45	0.002875844
B_5	0.003255973
R_2	0.007696904
D_46	0.003753165
D_47	0.001861653
D_48	0.004216627
D_49	0.005031512
B_6	0.001380175
B_7	0.003874053
B_8	0.001737927
D_50	0.005968964
D_51	0.001183048
B_9	0.015268841
R_3	0.003481747
D_52	0.001647869
P_3	0.001883616
B_10	0.001368278
D_53	0.001750505
S_5	0.001164693
B_11	0.008066418
S_6	0.001524821
D_54	0.001795186

Feature_Custom	Feature_Importance
P_2	0.09675074
D_39	0.002307458
B_1	0.001583478
B_2	0.002885434
R_1	0.004235974
S_3	0.003366762
D_41	0.003626543
B_3	0.002314003
D_42	0.002310635
D_43	0.001857
D_44	0.001958768
B_4	0.003914777
D_45	0.001728525
B_5	0.00140152
R_2	0.003012407
D_46	0.002242335
D_47	0.002230299
D_48	0.02976977
D_49	0.005893676
B_6	0.002055588
B_7	0.00222936
B_8	0.00145471
D_50	0.002795371
D_51	0.00135725
B_9	0.006206631
R_3	0.002527386
D_52	0.001807333
P_3	0.002992001
B_10	0.001859464
D_53	0.00210789
S_5	0.001293326
B_11	0.014690178
S_6	0.001708343
D_54	0.001734754

Slide #7. XGBoost - Grid Search

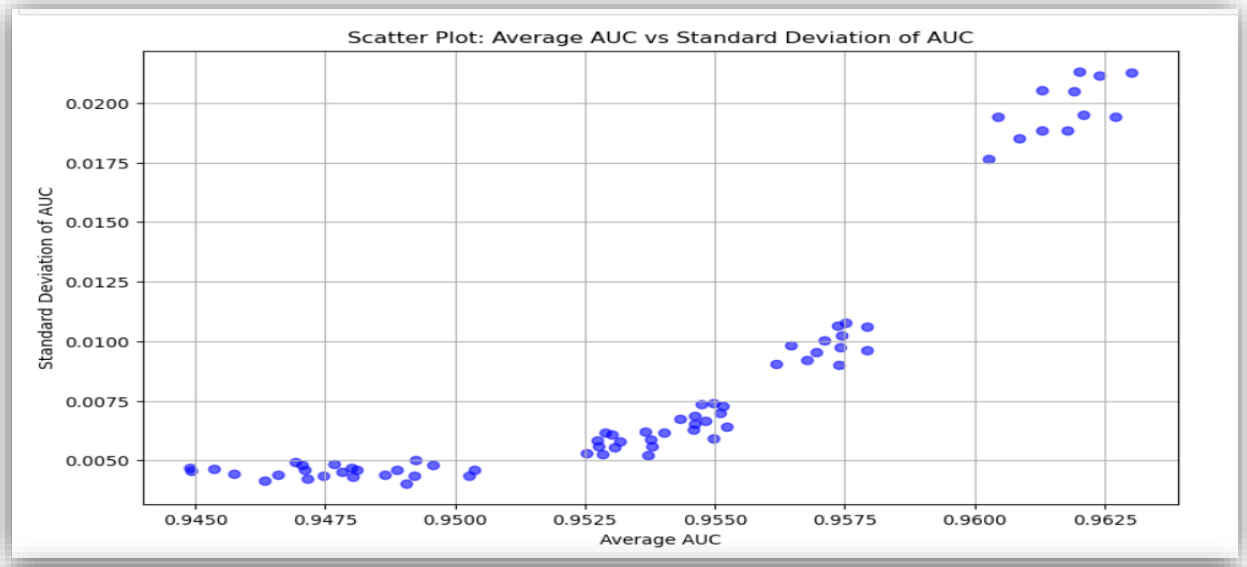
- **Objective**: Fine-tune the XGBoost model by testing multiple hyperparameter combinations to find the best configuration.
- **Parameter Selection**: We selected these parameters to balance model complexity and generalization. A lower learning rate helps the model converge more smoothly, while experimenting with the number of trees controls how much the model can learn. Subsample ratios and feature percentages prevent overfitting by introducing randomness, ensuring the model generalizes well across unseen data..
- **Experience**: The process was demanding, with *72* combinations (3 × 2 × 2 × 2 × 3) taking significant time to complete.
- **Lessons Learned**: Incremental saving was crucial to avoid losing progress. Two separate grid searches were conducted—one for XGBoost and one for the Neural Network.

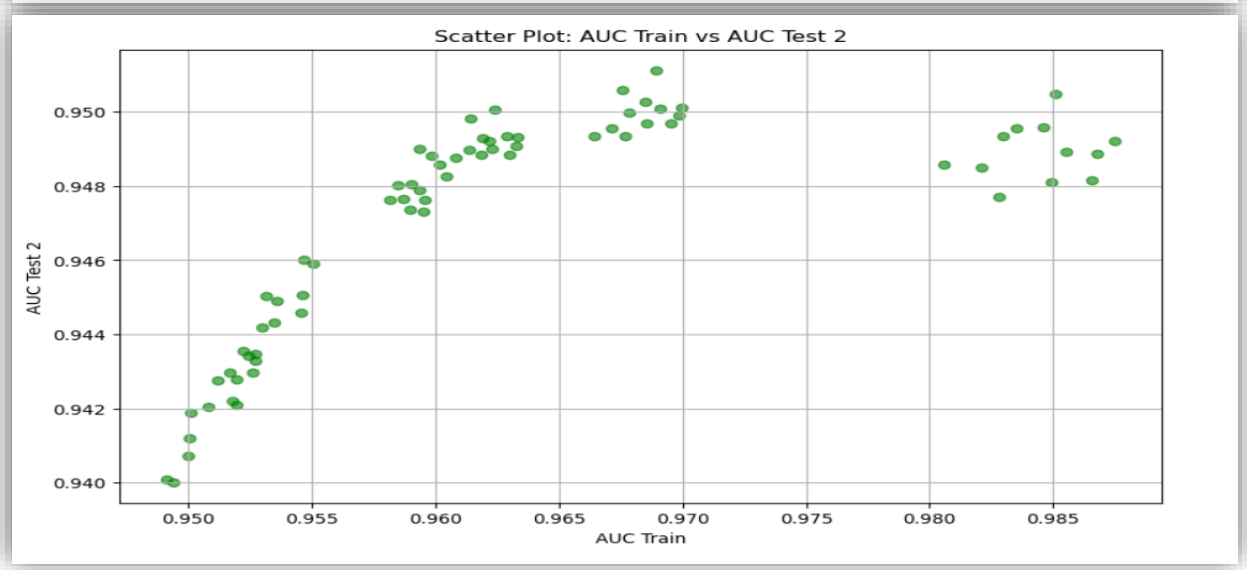
```python
# Initialize an empty list to store the results
results_list = []

# Step 3: Loop over all parameter combinations
for n in n_estimators:
    for lr in learning_rates:
        for subsample in subsamples:
            for colsample in colsample_bytrees:
                for weight in weights:
                    # Train the XGBoost model with the current combination
                    model = xgb.XGBClassifier(
                        n_estimators=n,
                        learning_rate=lr,
                        subsample=subsample,
                        colsample_bytree=colsample,
                        scale_pos_weight=weight,
                        use_label_encoder=False,
                        eval_metric='logloss'
                    )

                    # Fit the model on the train data
                    model.fit(X_train, Y_train)

                    # Calculate AUC for Train, Test1, and Test2 datasets
                    auc_train = roc_auc_score(Y_train, model.predict_proba(X_train)[:, 1])
                    auc_test1 = roc_auc_score(Y_test_1, model.predict_proba(X_test_1)[:, 1])
                    auc_test2 = roc_auc_score(Y_test_2, model.predict_proba(X_test_2)[:, 1])

                    # Store the results as a dictionary in the results_list
                    results_list.append({
                        '# Trees': n,
                        'LR': lr,
                        'Subsample': subsample,
                        '% Features': colsample,
                        'Weight of Default': weight,
                        'AUC Train': auc_train,
                        'AUC Test 1': auc_test1,
                        'AUC Test 2': auc_test2
                    })

                    # Print progress
                    print(f"Completed: Trees={n}, LR={lr}, Subsample={subsample}, "
                          f"Features={colsample}, Weight={weight}")

# Step 4: Convert the results list into a DataFrame
results = pd.DataFrame(results_list)
```

Slide #8. XGBoost - Grid Search Scatter Plots

Grid Search Scatter Plots

- **Model Selection Process**: From the scatter plots, we identified the optimal model with consistent performance, prioritizing both high AUC values and low variance across train and test samples.
- **Chosen Model Parameters**: The selected model has **100 trees**, a **learning rate of 0.1**, a **subsample ratio of 0.5**, and **50% feature usage**, balancing complexity and generalization.
- **Performance Metrics**: This model achieved an **AUC of 0.953** on Test 1 and **0.951** on Test 2, demonstrating robust predictive power across datasets with minimal overfitting.
- **Reason for Selection**: This model offered the best trade-off between train and test performance, aligning closely on both metrics while maintaining high predictive accuracy and stability.



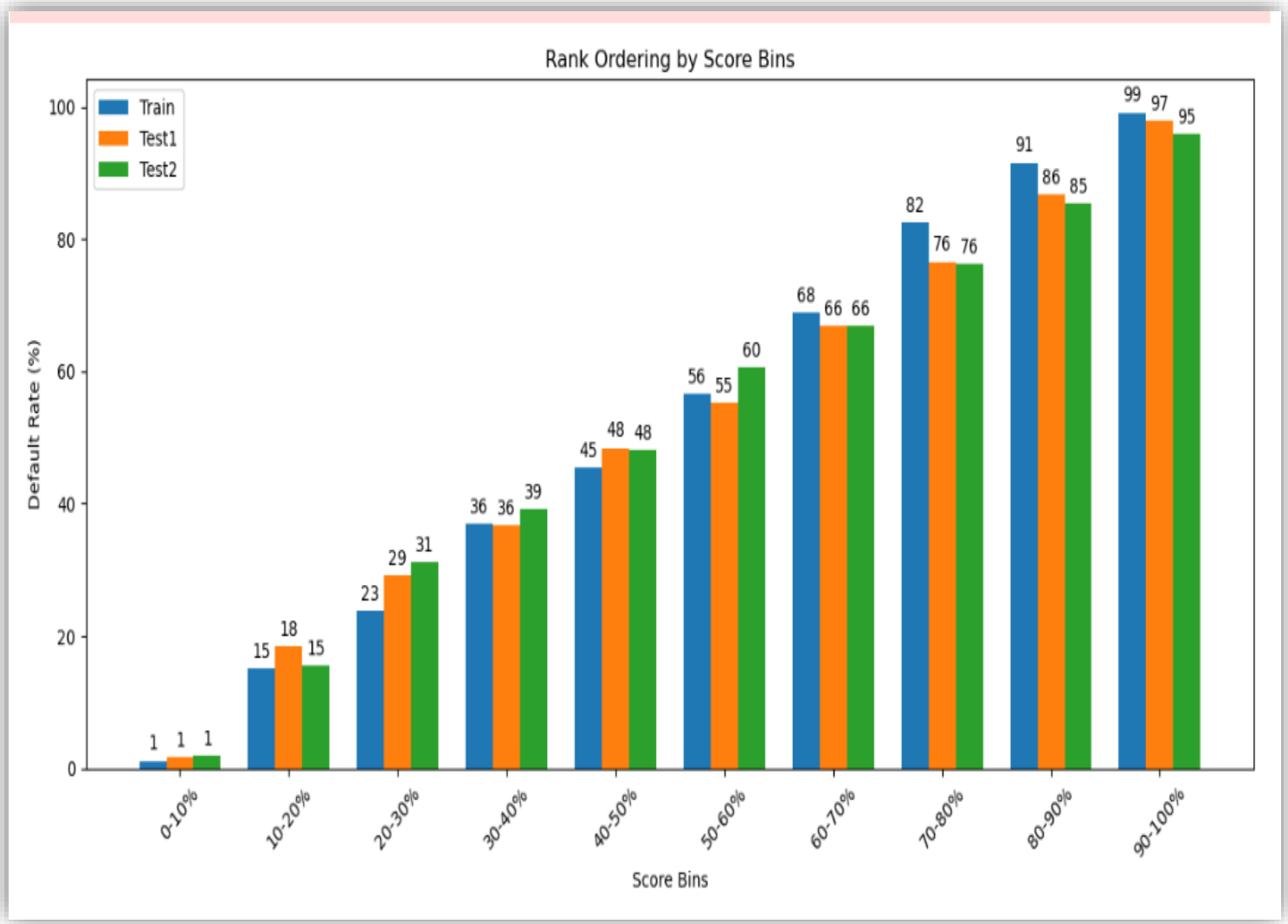


Slide #9. XGBoost – Final Model

- **Final Model Parameters**:
  The optimized XGBoost model used specific parameters to enhance performance.
  - •Trees: 100, Learning Rate: 0.1
  - •Subsample: 0.5, Feature %: 50%, Default Weight: 1

- **Model AUC Performance**:
  The model showed consistent AUC performance across Train, Test 1, and Test 2 datasets.
  - •Train: 0.968, Test 1: 0.954, Test 2: 0.951

Score Binning & Rank Ordering:
Score bins from the Train sample were used across all datasets, ensuring consistent risk differentiation.
- •Top 10% bin: Train (1%), Test 1 (1%), Test 2 (1%)
- •80-90% bin: Train (91%), Test 1 (87%), Test 2 (85%)



Slide #10. XGBoost – SHAP Analysis

- **Ranking of Attributes**:

P_2 has the highest mean SHAP value (~1.01), indicating it has the most significant impact on the model's output. S_3_Ave_6 and D_48 are the next most influential features, reflecting the importance of customer spending and delinquency history.

- **Beeswarm Graph Interpretation**:

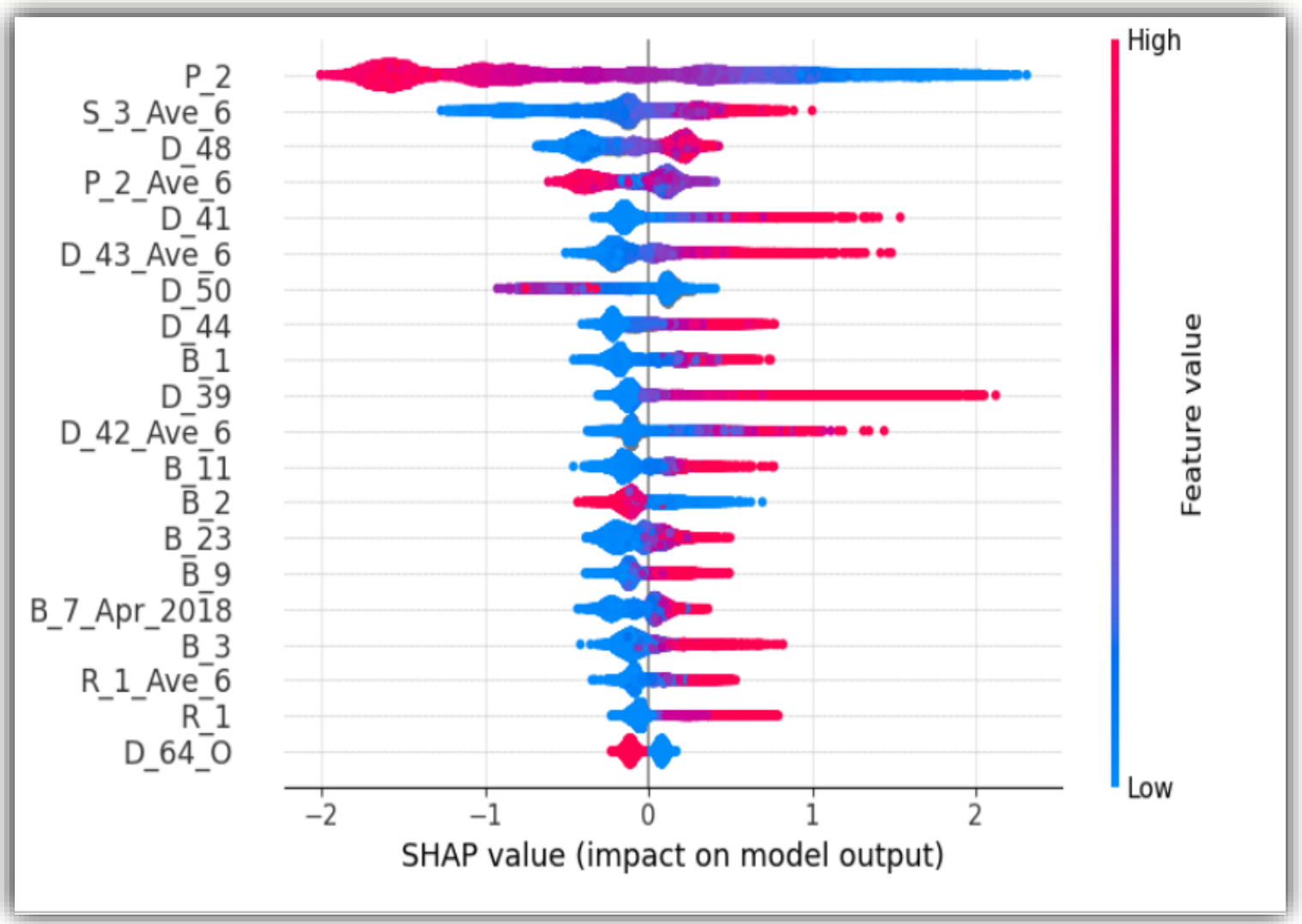
Each dot in the beeswarm graph represents an observation, with color showing the feature value (blue = low, red = high).

The spread of SHAP values helps visualize how much a feature's contribution varies across observations.

- **Positive and Negative Impacts**:

Features such as P_2 and D_48 show both positive and negative impacts on predictions. Higher values tend to push the prediction towards default (positive SHAP values).

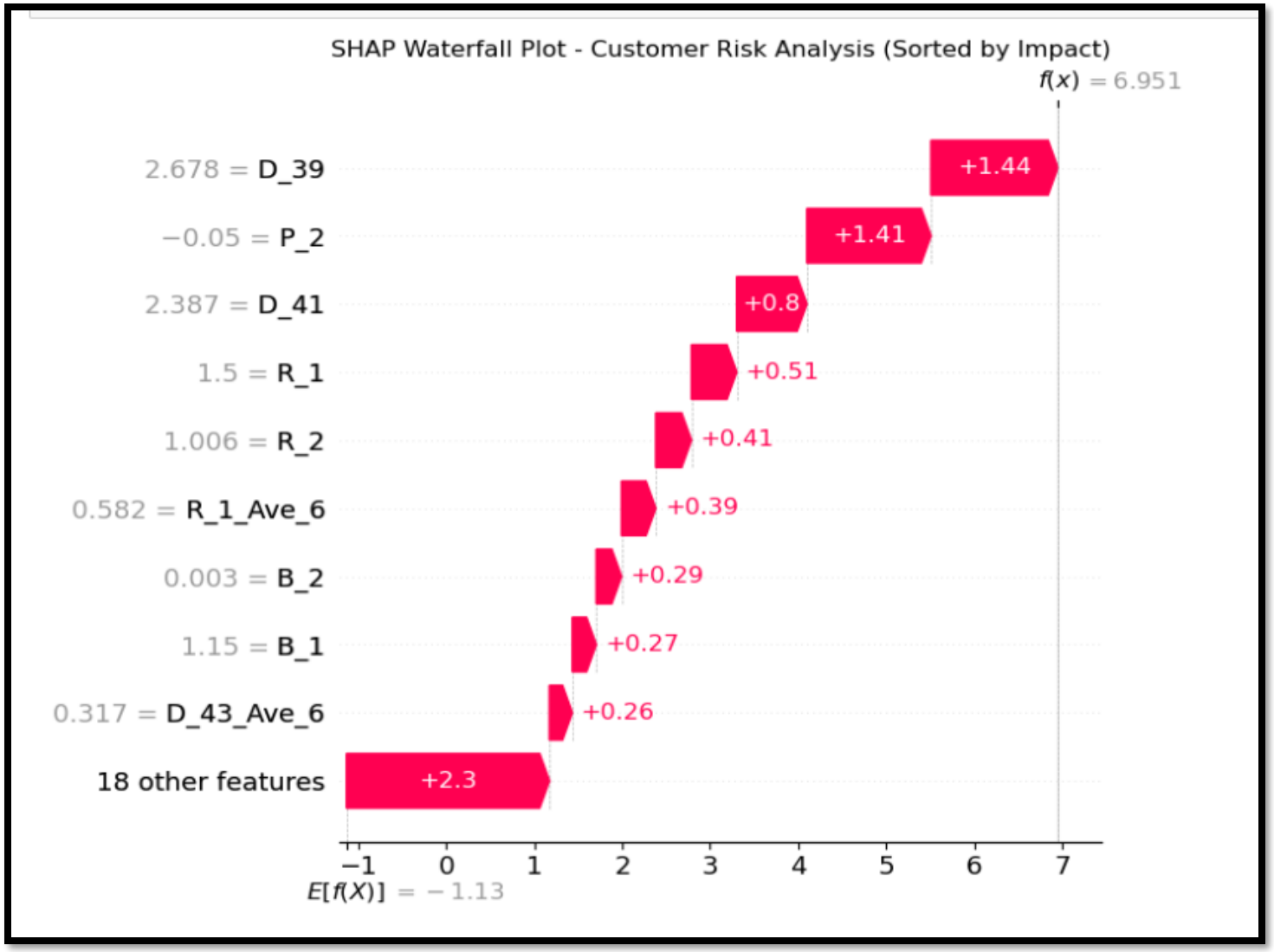
- Correlation Insight:For example, high values of S_3_Ave_6 (in red) are associated with increased default risk.Some features (like B_2) show that both high and low values contribute to risk, indicating non-linear relationships.



Slide #11. XGBoost – SHAP Analysis

Waterfall Plot for One Observation

- Overview of SHAP Waterfall Plot: The waterfall plot shows the contribution of each feature toward the final prediction for one specific customer. It highlights which features increase or decrease the model's output score.
- Key Drivers of the Score:D_39: This feature has the highest positive impact (+2.68) on pushing the score toward default risk.D_43_Ave_6: Adds another significant push (+2.3) toward a higher score, suggesting delinquency behavior.
- Negative Contributions:P_2: This feature reduces the risk slightly with a negative impact of -0.05, implying that better payment behavior reduces the default score.
- Cumulative Impact:The starting expected value is E[f(x)] = -1.13. After adding the impact of all features, the final score reaches 6.95, indicating a high probability of default for this customer.
- How to Improve the Score: Encouraging improvements in features with high positive SHAP values, such as reducing delinquency (D_39, D_41), could help lower the default risk for the customer.



Slide #12. Neural Network – Data Processing

- Handling Missing Values Step 1: Replaced missing values with 0 to maintain consistency across the dataset.Why: Neural networks cannot handle missing data, and replacing with 0 avoids errors during training without introducing bias..
- Outlier Treatment Step 2: Applied capping and flooring at the 1st and 99th percentiles to limit extreme values.Why: Outliers can negatively affect model learning by skewing predictions. This step ensures the model focuses on meaningful patterns.
- Feature Scaling with StandardScaler Step 3: Fitted StandardScaler on the training data and applied the same transformation to test datasets.Why: Neural networks are sensitive to feature magnitudes, and scaling ensures that all features contribute equally during training..
- Using Selected Features Only Step 4: All transformations were applied only to features selected in Step 10 to enhance the model's efficiency.Why: Focusing on the most relevant features improves performance by reducing noise and computational cost.

Summary of Data Processing Steps

Step	Description	Technique Used
Feature Selection	Selected Step 10 features	Feature importance from XGBoost
Missing Value Imputation	Replaced missing values with 0	.fillna(0)
Outlier Treatment	Capped at 1st and 99th percentiles	clip(...)
Normalization	Standardized features	StandardScaler(...)

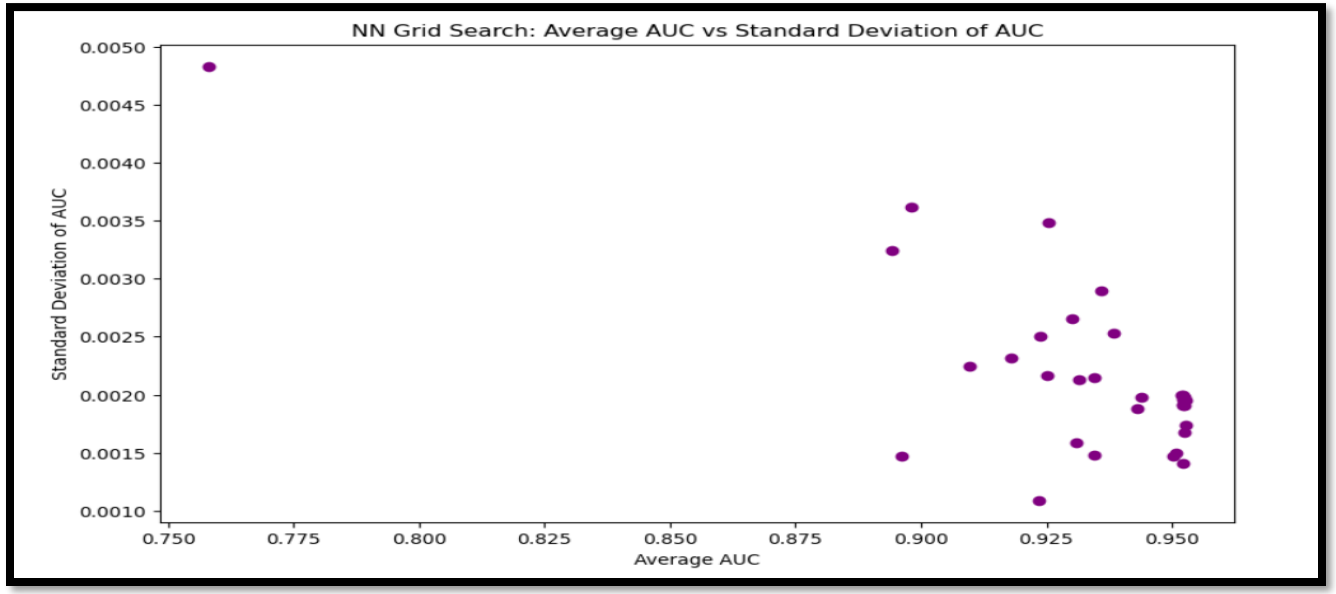
Slide #13. Neural Network - Grid Search

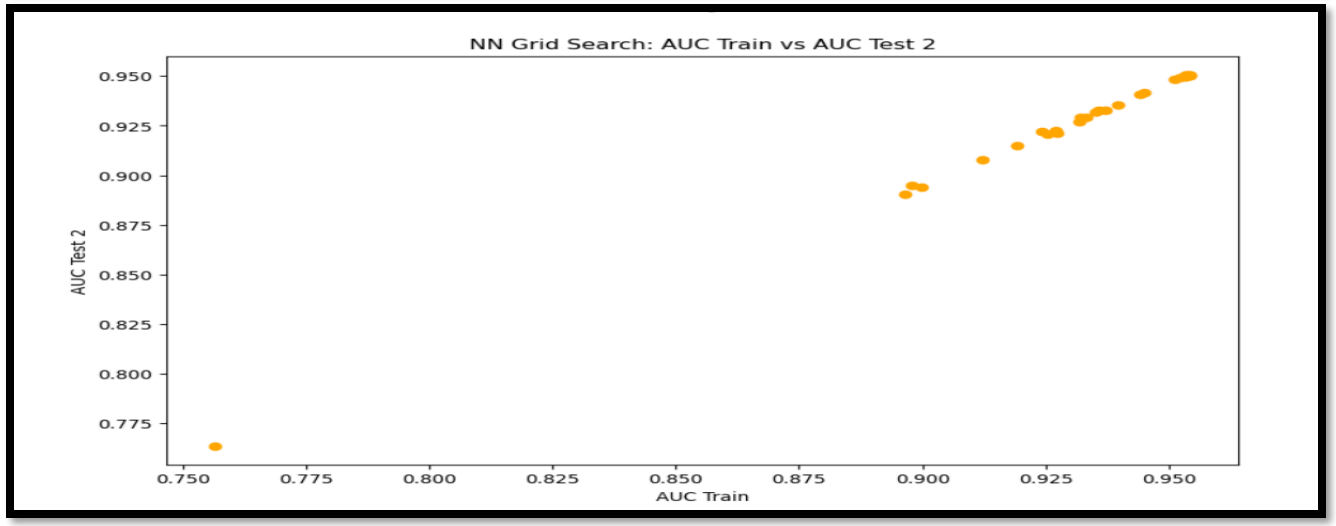
- Objective: Perform a grid search to identify the optimal hyperparameters for the Neural Network model using AUC scores across Train, Test1, and Test2 datasets..
- Hyperparameters Explored: Hidden Layers (2, 4), Nodes per Layer (4, 6), Activation Functions (ReLU, Tanh), Dropout (50%, 100%), Batch Sizes (100, 10,000). Total of **32** combinations tested..
- Model Configuration: Using Adam optimizer and Cross Entropy Loss, trained for 20 epochs with AUC as evaluation metric.
- Key Findings: Grid search proved computationally intensive, highlighting the importance of strategic parameter selection to prevent overfitting. Results were saved incrementally throughout the process.

```python
# Grid Search Loop
for hl in param_grid['hidden_layers']:
    for node in param_grid['nodes']:
        for activation in param_grid['activation']:
            for dropout in param_grid['dropout']:
                for batch_size in param_grid['batch_size']:
                    try:
                        model = create_model(hl, node, activation, dropout)
                        model.fit(X_train, Y_train, batch_size=batch_size, epochs=20, verbose=0)

                        # Evaluate the model
                        auc_train = roc_auc_score(Y_train, model.predict(X_train).ravel())
                        auc_test1 = roc_auc_score(Y_test_1, model.predict(X_test_1).ravel())
                        auc_test2 = roc_auc_score(Y_test_2, model.predict(X_test_2).ravel())

                        # Store the results using pd.concat
                        new_result = pd.DataFrame([{
                            '# HL': hl,
                            '# Node': node,
                            'Activation Function': activation,
                            'Dropout': f"{int(dropout * 100)}%",
                            'Batch Size': batch_size,
                            'AUC Train': auc_train,
                            'AUC Test 1': auc_test1,
                            'AUC Test 2': auc_test2
                        }])

                        results = pd.concat([results, new_result], ignore_index=True)

                        # Save results incrementally
                        results.to_csv('nn_grid_search_results.csv', index=False)

                        print(f"Completed: HL={hl}, Node={node}, Activation={activation}, "
                              f"Dropout={int(dropout * 100)}%, Batch={batch_size}")

                    except Exception as e:
                        print(f"Error with HL={hl}, Node={node}, Activation={activation}, "
                              f"Dropout={int(dropout * 100)}%, Batch={batch_size}: {e}")

print("Grid search completed and results saved to 'nn_grid_search_results.csv'.")
```

Slide #14. Neural Network - Grid Search

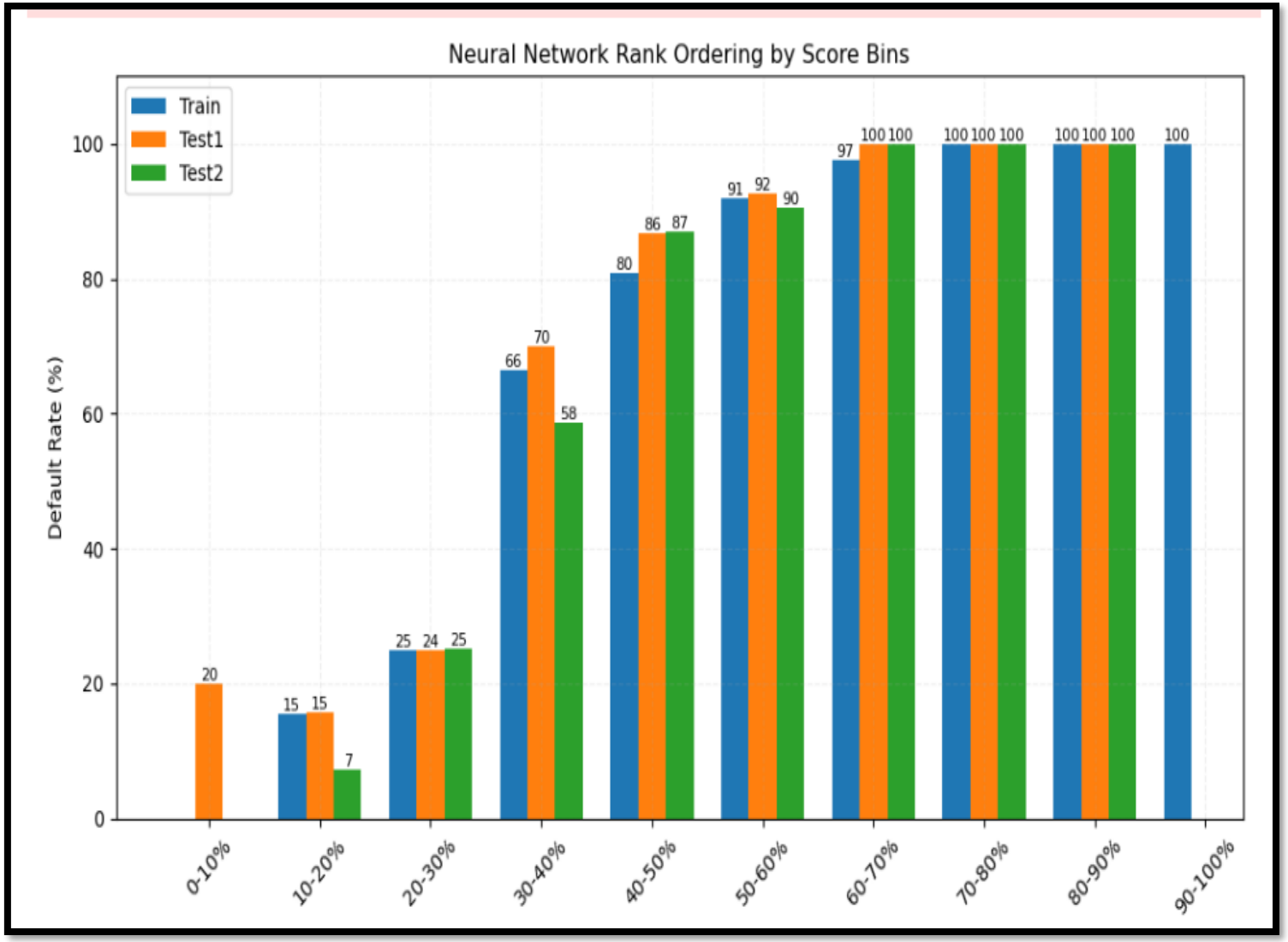
- Average AUC vs. Standard DeviationBest performing models show high AUC scores (0.93-0.95) with minimal standard deviation, clustering distinctly in bottom-right cornerSelected model prioritizes both prediction accuracy and performance stability across all evaluation datasets.
- Train vs. Test2 AUCModels positioned near diagonal line (AUC range 0.90-0.95) demonstrate consistent performance between training and testingChosen model achieves ~0.94-0.95 AUC for both datasets, indicating strong generalization without overfitting issues
- Key Takeaways Optimal model combines high average AUC scores, low performance variance, and balanced train-test metrics across all datasetsFinal results demonstrate robust predictive capabilities without overfitting, suitable for production deployment





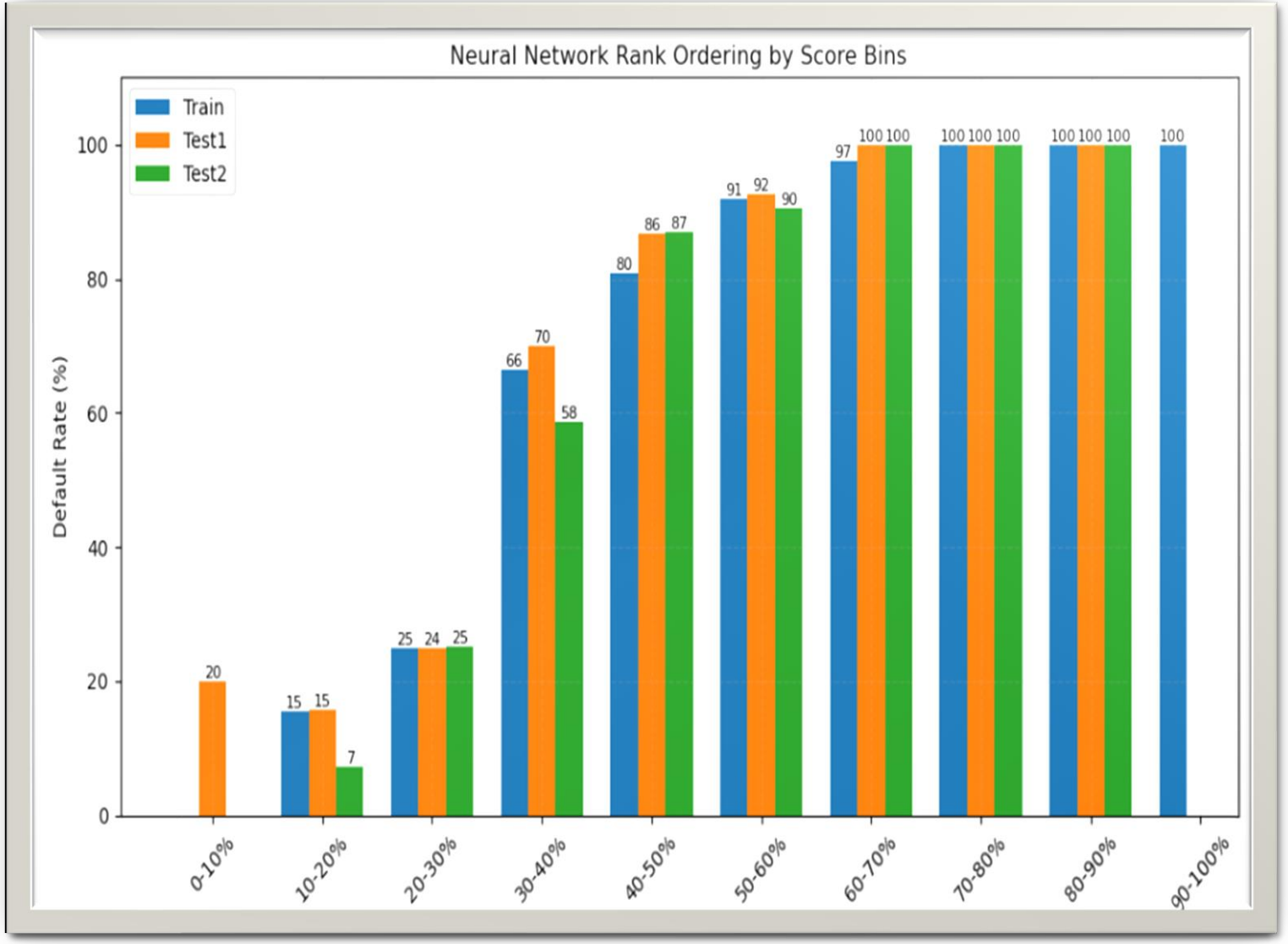
Slide 15. Neural Network - Grid Search

- Model Configuration 2 hidden layers, 6 nodes, ReLU activation. AUC ~0.95 across Train, Test1, and Test2 samples, showing consistent accuracy
- Rank Ordering Power Default rates increase steadily across score bins20% defaults in the lowest bin vs 100% in the highest bin. Similar rank ordering pattern across Train, Test1, and Test2
- Key Achievement and Comparison with XGBoost Provides reliable risk differentiation and predictive power While XGBoost achieved slightly higher AUC (~0.95-0.97), the neural network also generalizes well with strong rank ordering consistency, making it a viable alternative for production use.



Slide #16. Final Model

- **Performance Across Test Sets:** XGBoost achieved higher AUC scores (0.951 on Test 1, 0.947 on Test 2) compared to Neural Network (0.931 on Test 1, 0.928 on Test 2).
- **Consistency:** XGBoost maintained a smaller performance gap between the two test sets, indicating better generalizability.
- Separation Power:XGBoost achieves better separation in the lower bins. For instance, the default rate is only 1% in the 0-10% bin across all samples.Neural Network's separation is less consistent; for example, default rates in the 20-30% bin hover around 24%-25% but drop unexpectedly to 7% in Test2's 10-20% bin.
- **Recommendation**: XGBoost is preferred for its superior AUC and consistent performance across both test datasets, demonstrating better predictive power.



Slide #17. Strategy

	Train			Test 1			Test 2		
Threshold	Total	Default Rate	Revenue	Total	Default Rate	Revenue	Total	Default Rate	Revenue
0.1	38175	0.01124	209.59055	8201	0.01744	9.66783	8175	0.01859	9.08608
0.2	41628	0.02289	310.15886	8939	0.03110	13.47723	8885	0.02949	13.69142
0.3	44147	0.03513	397.54940	9467	0.04574	18.97395	9425	0.04562	18.65485
0.4	46314	0.05078	482.34334	9908	0.05995	22.55265	9968	0.06451	24.03439
0.5	48536	0.06925	578.90663	10395	0.07985	27.90561	10446	0.08357	30.20781
0.6	50807	0.09140	693.61978	10902	0.10191	31.73481	10943	0.10719	36.21570
0.7	53307	0.11952	825.70512	11454	0.12921	36.86619	11468	0.13289	41.43933
0.8	56239	0.15633	986.55971	12081	0.16216	45.12420	12112	0.16653	49.97036
0.9	59578	0.19892	1172.43794	12814	0.20275	54.90159	12804	0.20353	57.88748
1	64247	0.25651	1461.99836	13767	0.25648	70.80978	13768	0.25654	71.07133