Dynamic Data-Driven GitHub Copilot Prompts for DA5401 A3

Adaptive Insights and Dynamic Visualizations Based on Real Data Patterns

> **Key Principle**: Every visualization and insight must adapt to the actual data patterns discovered, not predetermined static outputs.

🎯 PROMPT 1: Smart Setup with Data-Aware Configuration

```python
# COPILOT PROMPT: Create intelligent setup that adapts configuration based on system capabilities and data siz
# REQUIREMENTS:
# - Detect available system memory and CPU cores to optimize processing
# - Auto-configure visualization settings based on data size (sample for large datasets)
# - Set up adaptive color palettes that change based on class imbalance severity
# - Create dynamic figure sizing based on screen resolution detection
# - Implement smart caching system for computationally expensive operations
# DYNAMIC BEHAVIOR: Configuration adapts to hardware, data size, and imbalance ratio
# EXAMPLE: If imbalance ratio > 100:1, use logarithmic scales; if < 10:1, use linear scales
```

🎯 PROMPT 2: Adaptive Data Loading with Intelligent Preprocessing Detection

```python
# COPILOT PROMPT: Create data loading system that automatically detects and adapts to dataset characteristics
# REQUIREMENTS:
# - Automatically detect if data needs scaling, normalization, or is already preprocessed
# - Identify feature types (PCA components, raw features, categorical) and adapt analysis accordingly
# - Calculate dynamic memory usage and implement smart sampling for large datasets
# - Detect data quality issues and automatically suggest/apply corrections
# - Create adaptive data summary that focuses on most important characteristics discovered
# DYNAMIC BEHAVIOR:
# - If dataset > 1M rows, implement smart sampling with stratification
# - If features are PCA-transformed, skip certain analyses and adapt visualizations
# - If high correlation detected, automatically suggest feature selection
# ADAPTIVE INSIGHTS: Generate warnings and recommendations based on actual data characteristics
```

🎯 PROMPT 3: Dynamic Class Imbalance Analysis with Adaptive Insights

```python
```

```
# COPILOT PROMPT: Create class imbalance analysis that generates insights based on actual imbalance severity
# REQUIREMENTS:
# - Calculate imbalance ratio and automatically categorize severity (mild/moderate/severe/extreme)
# - Generate different visualizations based on imbalance level:
#   * Mild (2:1 to 10:1): Standard bar/pie charts
#   * Moderate (10:1 to 100:1): Logarithmic scales with zoom insets
#   * Severe (100:1 to 1000:1): Nested pie charts with magnified minority segment
#   * Extreme (>1000:1): Specialized visualizations with proportional area representations
# - Auto-generate business impact statements based on actual imbalance ratio
# - Create dynamic recommendations for resampling approaches based on imbalance severity
# ADAPTIVE INSIGHTS:
# - "With X:1 imbalance, traditional accuracy will be Y% misleading"
# - "At this imbalance level, expect Z% precision drop without resampling"
# - Auto-calculate business costs based on actual class distribution
# DYNAMIC BEHAVIOR: Visualization complexity and insight depth scale with imbalance severity
```

🎯 PROMPT 4: Dynamic Class Imbalance Analysis with Adaptive Insights

python

```python
# COPILOT PROMPT: Create smart train-test split that adapts strategy based on dataset size and class distributio
# REQUIREMENTS:
# - Calculate optimal test size based on minority class count (minimum 30 minority samples in test)
# - Implement stratified sampling with adaptive strata based on feature distributions
# - Auto-detect if dataset allows for validation set creation
# - Generate split quality metrics that adapt to actual class distributions
# - Create adaptive visualizations showing split effectiveness
# DYNAMIC BEHAVIOR:
# - If minority class < 100 samples, use larger training proportion (90/10 instead of 80/20)
# - If high dimensionality, implement additional validation for split representativeness
# - Auto-adjust random_state if initial split creates poor stratification
# ADAPTIVE INSIGHTS:
# - "With X minority samples, Y test samples provide Z% confidence in evaluation"
# - "Split quality score: A/100 based on feature distribution preservation"
# - Auto-generate warnings if split quality is suboptimal
```

🎯 PROMPT 5: Dynamic Baseline Model with Performance-Adaptive Evaluation

python

```
# COPILOT PROMPT: Create baseline model evaluation that adapts metrics and visualizations to actual performar
# REQUIREMENTS:
# - Automatically select most informative metrics based on model performance characteristics
# - Create adaptive confusion matrix visualization based on prediction patterns:
#   * High precision, low recall: Focus on missed fraud cases
#   * Low precision, high recall: Focus on false alarm analysis
#   * Balanced performance: Standard visualization
#   * Poor performance: Add diagnostic plots for failure analysis
# - Generate performance-specific insights and improvement recommendations
# - Create adaptive ROC/PR curves with dynamic zoom regions based on operating point
# ADAPTIVE INSIGHTS:
# - If precision < 0.1: "Model generates X false alarms per real fraud - consider cost implications"
# - If recall < 0.3: "Model misses Y% of fraud cases - investigate feature adequacy"
# - Auto-calculate business metrics: "At current performance, expect $Z daily false alarm costs"
# DYNAMIC BEHAVIOR: Evaluation depth and visualization complexity adapt to performance patterns
```

🎯 PROMPT 6: Intelligent SMOTE with Data-Pattern Adaptive Implementation

python

```python
# COPILOT PROMPT: Create SMOTE implementation that adapts parameters and analysis based on data geometr
# REQUIREMENTS:
# - Auto-detect optimal k_neighbors based on minority class density and dimensionality
# - Analyze local data density and adapt SMOTE strategy accordingly:
#   * High density regions: Standard SMOTE
#   * Low density regions: BorderlineSMOTE or ADASYN
#   * Mixed density: Adaptive k per region
# - Create visualization that shows actual synthetic sample distribution, not generic examples
# - Implement quality assessment of generated samples with adaptive thresholds
# - Generate insights based on actual synthetic sample characteristics
# ADAPTIVE INSIGHTS:
# - "SMOTE generated X samples in high-density regions, Y in sparse areas"
# - "Synthetic sample quality score: Z% based on nearest neighbor analysis"
# - "Recommended k_neighbors: A (based on local intrinsic dimensionality)"
# DYNAMIC BEHAVIOR:
# - Visualization adapts to actual sample distribution patterns found
# - Quality metrics adapt to dimensionality and density characteristics
# - Parameter recommendations based on discovered data geometry
```

🎯 PROMPT 7: Adaptive Clustering-Based Oversampling with Data-Driven Cluster Analysis

python

```python
# COPILOT PROMPT: Create CBO system that determines optimal clustering strategy based on actual minority cl
# REQUIREMENTS:
# - Implement multiple clustering validation metrics (silhouette, calinski-harabasz, davies-bouldin)
# - Auto-detect optimal k using multiple methods and select based on data characteristics:
#   * Use silhouette for well-separated data
#   * Use elbow method for overlapping clusters
#   * Use gap statistic for unclear structure
# - Analyze actual cluster characteristics and adapt oversampling strategy:
#   * Tight clusters: Generate samples near centroids
#   * Loose clusters: Generate samples throughout cluster space
#   * Irregular clusters: Use density-based sampling
# - Create visualizations that show actual discovered cluster patterns
# - Generate insights based on real cluster analysis results
# ADAPTIVE INSIGHTS:
# - "Discovered X distinct fraud patterns with Y% separation quality"
# - "Cluster A represents Z% of fraud cases with characteristics: [actual pattern description]"
# - "Recommended oversampling: A samples from tight clusters, B from loose clusters"
# DYNAMIC BEHAVIOR:
# - Clustering algorithm selection adapts to data characteristics
# - Oversampling strategy adapts to actual cluster properties found
# - Visualizations adapt to cluster count and separation quality
```

🎯 PROMPT 8: Smart Clustering-Based Undersampling with Adaptive Strategy Selection

python

```
# COPILOT PROMPT: Create CBU system that selects undersampling strategy based on majority class structure a
# REQUIREMENTS:
# - Analyze majority class distribution and automatically select optimal undersampling approach:
#   * Uniform distribution: Random proportional sampling
#   * Clustered distribution: Cluster-aware sampling
#   * Near minority boundary: Distance-based removal
#   * Mixed patterns: Hybrid approach
# - Calculate actual distance metrics to minority class and use for intelligent sample selection
# - Implement multiple undersampling strategies and auto-select based on data geometry
# - Generate insights about majority class patterns discovered
# - Create adaptive visualizations showing actual removal strategy effectiveness
# ADAPTIVE INSIGHTS:
# - "Majority class shows X distinct patterns, removing Y% from each"
# - "Distance analysis: Z% of majority samples within boundary region"
# - "Undersampling preserved A% of original majority class diversity"
# DYNAMIC BEHAVIOR:
# - Strategy selection adapts to discovered majority class structure
# - Sample removal adapts to actual proximity patterns
# - Quality metrics adapt to preservation of class characteristics
```

🎯 PROMPT 9: Performance-Adaptive Model Training with Real-time Optimization

```python
# COPILOT PROMPT: Create model training system that adapts parameters based on data characteristics and mo
# REQUIREMENTS:
# - Auto-detect if logistic regression needs regularization based on feature characteristics
# - Implement adaptive solver selection based on dataset size and condition number
# - Monitor convergence and automatically adjust max_iter if needed
# - Track actual training metrics and generate insights about model behavior
# - Implement early stopping if performance plateaus
# ADAPTIVE INSIGHTS:
# - "Model converged in X iterations, suggesting Y about data separability"
# - "Regularization parameter C=Z optimal based on cross-validation performance"
# - "Training set size effect: A% performance gain from resampling"
# DYNAMIC BEHAVIOR:
# - Training parameters adapt to actual data characteristics
# - Monitoring adapts to convergence patterns observed
# - Insights generated based on actual training behavior
```

🎯 PROMPT 10: Dynamic Model Comparison with Adaptive Performance Analysis

```python
```

```
# COPILOT PROMPT: Create model comparison system that adapts analysis depth based on performance differe
# REQUIREMENTS:
# - Calculate statistical significance of performance differences and adapt visualization accordingly:
#   * Large differences: Standard comparison charts
#   * Small differences: Confidence intervals and significance tests
#   * No significant differences: Focus on computational efficiency and interpretability
# - Generate adaptive insights based on actual performance patterns discovered:
#   * Clear winner: Focus on why it performs better
#   * Close competition: Analyze trade-offs and business implications
#   * All poor: Diagnostic analysis and recommendations
# - Create visualizations that emphasize most important differences found
# - Implement adaptive metric selection based on business impact patterns
# ADAPTIVE INSIGHTS:
# - If one model significantly better: "Model X shows Y% improvement with Z confidence"
# - If models similar: "Performance differences not significant (p=A), choose based on B criteria"
# - If all poor: "All models show C limitation, suggest D alternative approaches"
# DYNAMIC BEHAVIOR:
# - Analysis depth adapts to significance of performance differences
# - Visualization emphasis adapts to most important findings
# - Recommendations adapt to actual performance patterns discovered
```

🎯 PROMPT 11: Intelligent Business Impact Analysis with Real Data-Driven Recommendations

```python
# COPILOT PROMPT: Create business impact analysis that uses actual model performance to calculate real busin
# REQUIREMENTS:
# - Calculate actual financial impact using discovered performance metrics:
#   * Use actual precision/recall to estimate false alarm costs
#   * Use actual detection rates to estimate fraud prevention savings
#   * Factor in actual class distribution for realistic projections
# - Generate ROI calculations based on real performance differences between models
# - Create adaptive recommendations based on actual business impact calculations
# - Implement sensitivity analysis using actual performance uncertainty
# ADAPTIVE INSIGHTS:
# - "Model X would prevent $Y fraud annually with $Z false alarm costs (based on actual performance)"
# - "Break-even point: Model becomes profitable when fraud rate exceeds A% (calculated from actual metrics)"
# - "Recommended model: B provides $C net benefit under current fraud patterns"
# DYNAMIC BEHAVIOR:
# - All calculations use actual discovered performance metrics
# - Recommendations adapt to real cost-benefit analysis results
# - Sensitivity analysis based on actual performance confidence intervals
```

🎯 PROMPT 12: Adaptive Feature Importance and Pattern Discovery

```python
# COPILOT PROMPT: Create feature analysis that discovers actual patterns in model behavior and data
# REQUIREMENTS:
# - Analyze actual feature importance patterns across all models
# - Discover which PCA components are most discriminative for fraud detection
# - Identify actual decision boundaries and visualize them
# - Find real patterns in misclassified samples and generate insights
# - Create adaptive visualizations based on discovered importance patterns
# ADAPTIVE INSIGHTS:
# - "Component V4 shows highest discriminative power across all models"
# - "Misclassification pattern: X% of errors occur in [actual pattern description]"
# - "Decision boundary analysis reveals Y overlapping regions"
# DYNAMIC BEHAVIOR:
# - Analysis adapts to actual feature importance discovered
# - Visualizations emphasize most important patterns found
# - Insights generated from real pattern discovery, not assumptions
```

🎯 PROMPT 13: Real-time Diagnostic System with Adaptive Problem Detection

```python
# COPILOT PROMPT: Create diagnostic system that automatically detects issues based on actual analysis results
# REQUIREMENTS:
# - Monitor all analysis steps for potential issues and automatically flag problems:
#   * Data quality issues discovered during loading
#   * Clustering instability detected during resampling
#   * Model convergence problems during training
#   * Performance anomalies during evaluation
# - Generate specific diagnostic insights based on actual problems found
# - Provide adaptive solutions based on detected issue types
# - Create alerts with varying severity based on actual impact assessment
# ADAPTIVE INSIGHTS:
# - If clustering unstable: "Clustering shows X% variance across runs, consider Y approach"
# - If performance anomalies: "Model Z shows unexpected behavior: [specific pattern description]"
# - If data issues: "Detected A anomaly affecting B% of samples"
# DYNAMIC BEHAVIOR:
# - Diagnostic depth adapts to severity of issues found
# - Solutions recommended based on actual problem characteristics
# - Alert priority based on actual impact assessment
```

🎯 PROMPT 14: Adaptive Visualization Engine with Dynamic Chart Selection

```python
```

```
# COPILOT PROMPT: Create visualization system that automatically selects optimal chart types based on data pa
# REQUIREMENTS:
# - Analyze data characteristics and automatically choose best visualization approach:
#   * High dimensionality: Dimensionality reduction with explained variance
#   * Temporal patterns: Time series analysis with trend detection
#   * Cluster patterns: Appropriate cluster visualization with quality metrics
#   * Performance patterns: Optimal comparison visualization based on differences
# - Implement adaptive scaling, coloring, and annotation based on actual data ranges
# - Generate insights embedded directly in visualizations based on patterns discovered
# - Create interactive elements that adapt to data complexity
# DYNAMIC BEHAVIOR:
# - Chart types selected based on actual data characteristics
# - Scaling and ranges adapt to actual value distributions
# - Annotations generated from real pattern analysis
# - Interactivity level adapts to data complexity
```

🎯 PROMPT 15: Master Integration with Adaptive Workflow Orchestration

```python
# COPILOT PROMPT: Create master orchestration system that adapts workflow based on discovered data charac
# REQUIREMENTS:
# - Analyze initial data characteristics and adapt entire analysis workflow:
#   * Small datasets: Run all analyses without sampling
#   * Large datasets: Implement smart sampling strategies
#   * High-quality data: Focus on advanced modeling
#   * Poor-quality data: Emphasis on diagnostic and cleaning steps
# - Generate adaptive final report that emphasizes most important findings discovered
# - Create executive summary that adapts content based on actual business impact calculated
# - Implement adaptive export formats based on intended audience and findings importance
# ADAPTIVE INSIGHTS:
# - Executive summary adapts content based on actual ROI calculations
# - Technical details included based on complexity of patterns discovered
# - Recommendations prioritized based on actual impact analysis
# DYNAMIC BEHAVIOR:
# - Workflow adapts to data characteristics discovered
# - Report structure adapts to importance of findings
# - Export formats adapt to audience needs and content complexity
```

🔄 DYNAMIC INSIGHT GENERATION PATTERNS

Pattern 1: Conditional Insights Based on Actual Values

```python
```

```python
# EXAMPLE: Instead of static text, generate insights like:
if imbalance_ratio > 100:
    insight = f"With {imbalance_ratio:.1f}:1 imbalance, accuracy will be {accuracy_misleading_factor:.1f}% misleadir
elif imbalance_ratio > 10:
    insight = f"Moderate imbalance of {imbalance_ratio:.1f}:1 requires careful metric selection"
else:
    insight = f"Relatively balanced at {imbalance_ratio:.1f}:1, standard metrics applicable"
```

Pattern 2: Performance-Adaptive Analysis

```python
# EXAMPLE: Analysis depth adapts to performance differences found
if max(f1_scores) - min(f1_scores) > 0.1:
    analysis_depth = "detailed_comparison"  # Large differences need deep analysis
elif statistical_significance < 0.05:
    analysis_depth = "statistical_validation"  # Small but significant differences
else:
    analysis_depth = "practical_considerations"  # Focus on other factors
```

Pattern 3: Data-Driven Visualization Selection

```python
# EXAMPLE: Chart type adapts to actual data characteristics
if cluster_separation_quality > 0.7:
    viz_type = "standard_cluster_plot"  # Well-separated clusters
elif cluster_separation_quality > 0.3:
    viz_type = "enhanced_cluster_plot_with_boundaries"  # Overlapping clusters
else:
    viz_type = "density_based_visualization"  # Poor separation
```

🎯 Key Principles for Dynamic Implementation:

1. **Never Hard-Code Insights**: All text generated based on actual calculated values
2. **Adaptive Visualizations**: Chart types, scales, and annotations adapt to data patterns
3. **Performance-Driven Analysis**: Analysis depth adapts to significance of findings
4. **Real Business Impact**: All recommendations based on actual cost-benefit calculations
5. **Data-Aware Workflows**: Processing steps adapt to dataset characteristics

These prompts will generate truly dynamic, data-driven analyses where every insight, visualization, and recommendation adapts to the actual patterns discovered in your specific dataset.