ДОДАТОК А

Реалізації просунутих компонентів аналізу

```
15    class CategoricalOutlierDetectionComponent(ReportComponent):  vkiriushyn +1
38        def analyze(self):  vkiriushyn +1
57
58            for col in cat_cols:
59                non_null = df[col].dropna()
60                if len(non_null) == 0:
61                    continue
62
63                null_ratio = 1 - (len(non_null) / n_rows)
64                if null_ratio > self.min_null_ratio_to_warn:
65                    high_null_cols.append({"column": col, "null_ratio": round(null_ratio, 4)})
66
67                freq = non_null.value_counts(normalize=True)
68                unique_ratio = len(freq) / len(non_null)
69
70                if self.skip_high_cardinality and unique_ratio > self.cardinality_threshold:
71                    skipped_cols.append({"column": col, "unique_ratio": round(unique_ratio, 4)})
72                    continue
73
74                threshold = self.min_frequency_threshold
75                if self.adaptive_threshold:
76                    adaptive_min = max(0.001, 5 / len(non_null))
77                    threshold = min(threshold, adaptive_min)
78
79                rare_values = freq[freq < threshold].index
80                rare_mask = df[col].isin(rare_values)
81                outlier_masks[col] = rare_mask
82
83                if self.use_chi_square and len(cat_cols) > 1:
84                    for other_col in cat_cols:
85                        if other_col != col:
86                            contingency = pd.crosstab(df[col], df[other_col])
87                            if not contingency.empty:
88                                chi2, p, _, _ = chi2_contingency(contingency)
89                                if p < 0.05:
90                                    explanations.append({
```

Рисунок А.1 – Реалізація аналізу у компоненті Categorical Outlier Detection



```
71  class CompositeQualityScoreComponent(ReportComponent):  vkiriushyn +1
132         return float(entropy(pattern_counts))
133
134     def _compute_statistical_features(self, df) -> Dict[str, float]:  1 usage  vkiriushyn
135         numeric_df = df.select_dtypes(include=[np.number]).dropna()
136
137         if numeric_df.shape[1] == 0:
138             return {
139                 "mean_skewness": 0.0,
140                 "mean_kurtosis": 0.0,
141                 "outlier_ratio": 0.0,
142                 "feature_variance_cv": 0.0
143             }
144
145         sample_size = min(10000, len(numeric_df))
146         if sample_size < len(numeric_df):
147             numeric_df = numeric_df.sample(n=sample_size, random_state=42)
148
149         # Skewness and kurtosis
150         skewness = numeric_df.skew()
151         kurtosis = numeric_df.kurtosis()
152
153         # Outlier detection using IQR
154         outlier_ratios = []
155         for col in numeric_df.columns:
156             Q1, Q3 = numeric_df[col].quantile([0.25, 0.75])
157             IQR = Q3 - Q1
158             lower, upper = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
159             outliers = (numeric_df[col] < lower) | (numeric_df[col] > upper)
160             outlier_ratios.append(outliers.mean())
161
162         # Variance coefficient of variation
163         variances = numeric_df.var()
164         var_cv = variances.std() / (variances.mean() + 1e-10)
165
```

Рисунок А.2 – Реалізація аналізу у компоненті Composite Quality Score

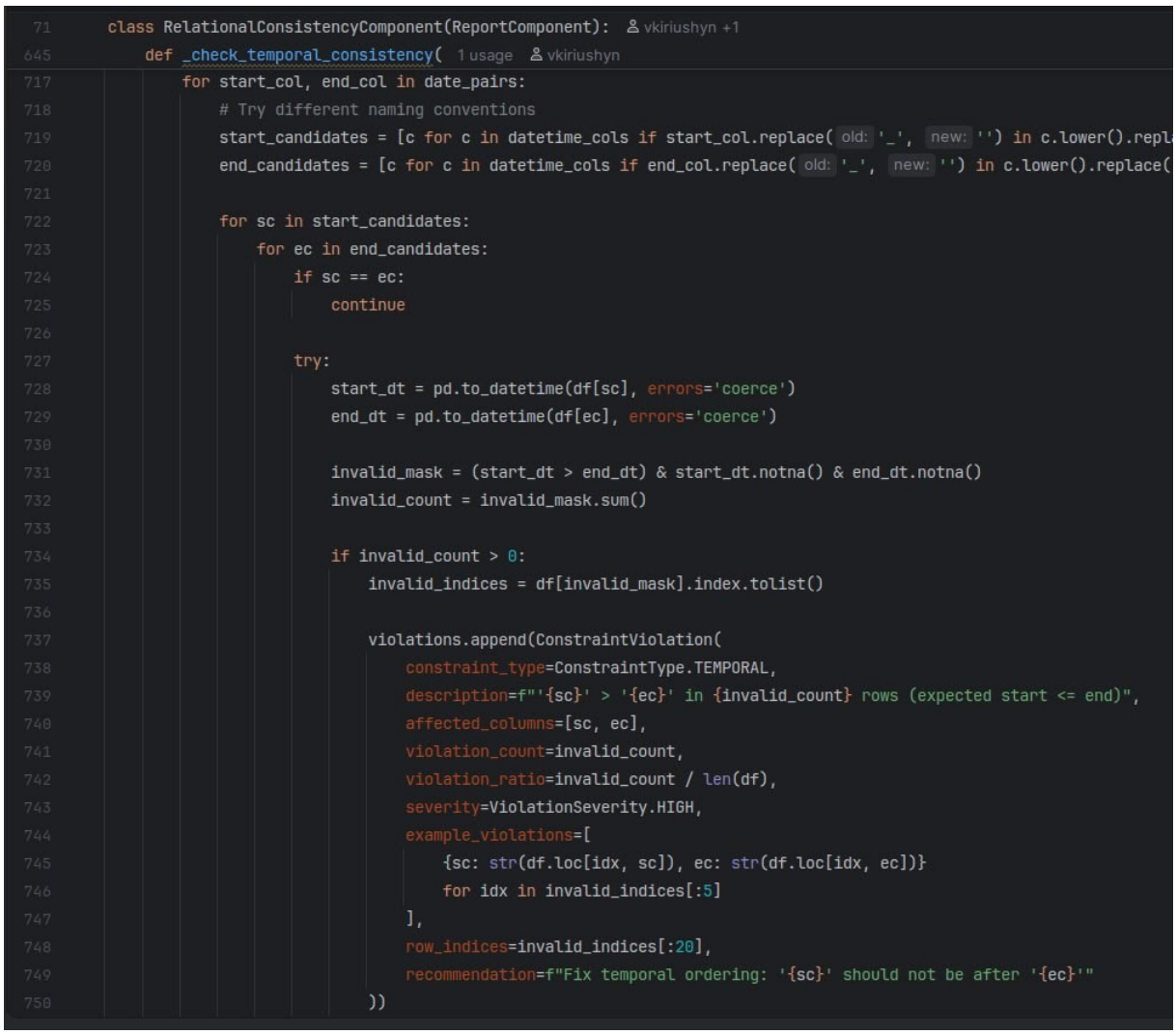
```
class LabelNoiseDetectionComponent(ReportComponent):  vkiriushyn +1
    def analyze(self) -> None:  vkiriushyn +1

        joint_sum = confident_joint.sum()
        if joint_sum > 0:
            estimated_joint = confident_joint / joint_sum
        else:
            estimated_joint = confident_joint.astype(float)

        metadata = {
            "n_samples": len(y),
            "n_classes": n_classes,
            "n_features": X_array.shape[1],
            "cv_folds": self.cv_folds,
            "confidence_threshold": self.confidence_threshold,
            "is_sampled": self._is_sampled,
            "original_size": self._original_size,
            "sample_size": len(y) if self._is_sampled else None,
            "is_large_dataset": is_large_dataset
        }

        self._result = NoiseAnalysisResult(
            noise_indices=noise_indices,
            noise_scores=noise_scores,
            noise_ratio=noise_ratio,
            noise_transition_matrix=transition_matrix,
            estimated_joint_distribution=estimated_joint,
            class_thresholds={str(labels[i]): float(thresholds[i]) for i in range(n_classes)},
            noise_type=noise_type,
            noise_type_confidence=type_confidence,
            per_class_noise_rates=per_class_noise,
            statistical_significance=stat_significance,
            ensemble_agreement=ensemble_agreement,
            metadata=metadata
        )
```

Рисунок А.3 – Реалізація аналізу у компоненті Label Noise Detection



```
class RelationalConsistencyComponent(ReportComponent):  vkiriushyn +1
    def _check_temporal_consistency(  1 usage  vkiriushyn
        for start_col, end_col in date_pairs:
            # Try different naming conventions
            start_candidates = [c for c in datetime_cols if start_col.replace( old: '_',  new: '') in c.lower().repl
            end_candidates = [c for c in datetime_cols if end_col.replace( old: '_',  new: '') in c.lower().replace(

            for sc in start_candidates:
                for ec in end_candidates:
                    if sc == ec:
                        continue

                    try:
                        start_dt = pd.to_datetime(df[sc], errors='coerce')
                        end_dt = pd.to_datetime(df[ec], errors='coerce')

                        invalid_mask = (start_dt > end_dt) & start_dt.notna() & end_dt.notna()
                        invalid_count = invalid_mask.sum()

                        if invalid_count > 0:
                            invalid_indices = df[invalid_mask].index.tolist()

                            violations.append(ConstraintViolation(
                                constraint_type=ConstraintType.TEMPORAL,
                                description=f"'{sc}' > '{ec}' in {invalid_count} rows (expected start <= end)",
                                affected_columns=[sc, ec],
                                violation_count=invalid_count,
                                violation_ratio=invalid_count / len(df),
                                severity=ViolationSeverity.HIGH,
                                example_violations=[
                                    {sc: str(df.loc[idx, sc]), ec: str(df.loc[idx, ec])}
                                    for idx in invalid_indices[:5]
                                ],
                                row_indices=invalid_indices[:20],
                                recommendation=f"Fix temporal ordering: '{sc}' should not be after '{ec}'"
                            ))
```

Рисунок А.4 – Реалізація аналізу у компоненті Relational Consistency

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	134 с.
	Інші документи	
2.	Презентаційні матеріали	23 плакати

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Кірюшин В.А.		05.05	Автоматизований аналіз якості даних на основі штучного інтелекту	Шифр групи	Код напр./спец.
Перевірив		Вітько О.В.				СШІм-24-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Чала Л.Е.					