REPORT STRUCTURE

1. Abstract

Key Points:

- One or two sentences of background: Kepler KOIs, automated vetting (Robovetter), `koi_score` as a quality/confidence score.
- Project objective: Classifying KOIs into high-quality vs low-quality classes based on a binary label derived from `koi_score` via thresholding. Using physical/observational features from the KOI table, apply PCA (via the `pca` Python library) and PyCaret's classification module to build a lightweight quality classifier, focusing on Logistic Regression, Random Forest, and MLP.
- Method overview:
  - Data source and cleaning;
  - Binary label construction using double thresholds (0.9 and 0.1 on `koi_score`), excluding mid-range samples;
  - Apply PCA using the `pca` library on standardized features;
  - PyCaret classification trained and compared in both Original feature space (`X_std`) and PCA feature space (`Z_pca`);
  - Focus on three algorithms: Logistic Regression, Random Forest, MLP.
- Main results (1–2 sentences):
  - Which model + feature combination performs best in terms of F1-score (e.g., "Random Forest achieves the highest F1 on original features, while Logistic Regression shows strong performance on PCA features");
  - Impact of PCA on classification performance/training stability.
- One sentence on significance:
  - This classifier can serve as an auxiliary KOI quality pre-screening tool, helping to quickly identify high-quality planet candidates for further analysis.

(Optional) Index Terms: Kepler Objects of Interest, koi_score, binary classification, principal component analysis, PyCaret, quality assessment, F1-

score

2. Introduction

> Corresponds to the high-level narrative of 1.1 Motivation & 1.2 Process & 1.3 Conclusion in FRAMEWORK.

Suggested subsections:

2.1 Scientific Background: Kepler KOIs and False Positives

- Brief description of the Kepler mission and KOIs: what are KOIs, sources of transit-like signals.
- Explain false positives: false signal problems caused by binary stars, background stars, noise, etc.
- Introduce automated vetting (Robovetter) and disposition (Candidate / False Positive).

2.2 The Role of `koi_score` as a Quality Metric

- Formally define `koi_score`: a continuous score from 0–1, representing the confidence/quality of vetting results.
- Explain its role in planet statistics/occurrence rate studies (e.g., commonly using threshold 0.9 to screen high-quality samples).
- Emphasize: `koi_score` is the output of a complex pipeline (numerous tests + Monte Carlo injection experiments), which ordinary users cannot easily reproduce.
- Note the bimodal distribution: `koi_score` exhibits two peaks near 0 and 1, making it well-suited for threshold-based binary classification.

2.3 Problem Statement and Objectives

- Describe from the perspective of "Advanced Statistical Approaches to Quality":
  - View `koi_score` as a "product/process quality metric," and KOI features as "process characteristics."
- Clearly state the problem definition:

    - Given physical and observational features in the KOI table, can we build a classification model that distinguishes high-quality KOIs (score ≥ 0.9) from low-quality KOIs (score ≤ 0.1)?
- Use 1–2 sentences to explain project objectives:
    - Construct a clean binary label from `koi_score` using double thresholds (0.9 and 0.1);
    - Build a binary classifier using PCA-transformed and original features;
    - Use PyCaret to systematically compare multiple classifiers, focusing on Logistic Regression, Random Forest, and MLP;
    - Analyze the impact of PCA on classification performance, stability, and interpretability;
    - Evaluate models primarily using F1-score, with Accuracy and ROC-AUC as secondary metrics.

2.4 Contributions and Report Organization

- List main contributions (bullets):
    - Build a complete pipeline based on the KOI table for binary quality classification using PCA + PyCaret;
    - Construct a well-defined binary label by applying double thresholds to `koi_score` and excluding ambiguous mid-range samples;
    - Systematically compare multiple classifiers (focusing on LR, RF, MLP) on original standardized features and PCA features;
    - Analyze the physical meaning of principal components and their relationship with KOI quality classification;
    - Demonstrate that F1-score effectively captures the balance between precision and recall for identifying high-quality KOIs.
- Outline the structure of subsequent sections (corresponding to each section).

---

3. Data Set and Preprocessing

> Corresponds to FRAMEWORK Step 1 (data acquisition, field selection, binary label construction, EDA).

Suggested subsections:

3.1 KOI Cumulative Table and Data Source

- Data source: NASA Exoplanet Archive KOI cumulative table (specify download method/version date).
- Total sample size (original ~9564), actual sample size after cleaning and applying thresholds.
- Briefly explain why this dataset is suitable for this course project (real engineering data, multivariate, appropriate size, bimodal `koi_score` distribution).

3.2 Column Groups and Feature Selection Strategy

- Explain the purpose of each column group according to FRAMEWORK logic:

1. **Identifiers (used only as labels, not in models):**

   `kepid`, `kepoi_name`, `kepler_name`, etc., used for labeling/tracing KOIs in analysis plots.

2. **Source for Binary Label Construction (not a feature):**

   `koi_score` (0–1) – used solely to derive the binary target variable; **not included as a predictor** to avoid label leakage.

3. **Classification Labels & False Positive Flags (used only for analysis/grouping, not in models):**

   - `koi_disposition`, `koi_pdisposition`
   - `koi_fpflag_nt`, `koi_fpflag_ss`, `koi_fpflag_co`, `koi_fpflag_ec`

     Explanation: They are highly correlated with `koi_score`. To avoid label leakage, they are only used for EDA and results discussion, not for PCA or classification input.

4. **Core Numerical Features for Models (12-dimensional feature vector X):**

   - Transit geometry & signal quality:

     `koi_period`, `koi_impact`, `koi_duration`, `koi_depth`, `koi_model_snr`

   - Planet properties & irradiation:

     `koi_prad`, `koi_teq`, `koi_insol`

   - Stellar properties:

`koi_steff`, `koi_slogg`, `koi_srad`

- Brightness: `koi_kepmag`
- **Excluded:** `ra`, `dec` (sky coordinates), `koi_time0bk` (transit epoch) – these do not carry physical information relevant to quality classification.
- Clearly specify pipeline metadata not used as features: `koi_tce_plnt_num`, `koi_tce_delivname`, etc., will be dropped.

5. **Uncertainty Columns (strategy for handling upper/lower error columns):**

   Clearly state: `*_err1`, `*_err2` (such as `koi_period_err1`, etc.) are not retained in this project baseline to avoid dimensional explosion and severe collinearity; only central value columns are used.

3.3 Construction of the Quality Label from `koi_score`

Add a dedicated subsection explaining the binary label construction:

- Describe the empirical distribution of `koi_score`:
  - The distribution shows two peaks near 0 and 1, indicating that most KOIs are confidently classified as either high-quality candidates or likely false positives.
- Explain the double-threshold approach:
  - **High-quality KOIs (label = 1):** `koi_score ≥ 0.9`
  - **Low-quality KOIs (label = 0):** `koi_score ≤ 0.1`
  - **Excluded samples:** All KOIs with `0.1 < koi_score < 0.9` are removed from the dataset
    - These mid-range samples are ambiguous and would introduce noise into the classification task
    - Removing them creates a cleaner, more well-defined binary classification problem
- Report the final dataset size after applying thresholds and filtering

3.4 Data Cleaning: Missing Values and Outliers

- Use `df.info()`, `df.describe()`, etc., to check for missing and anomalous values.
- Explain your strategy for handling missing values:

    - Remove rows missing `koi_score` (cannot construct label).
    - For the selected numerical astrophysical features (the 12-dimensional feature vector X), **do not** perform global mean/median imputation. Instead, **remove any samples that contain missing values in these features**.
    - Justify this choice briefly in the report: after applying the `koi_score` double-threshold (≥0.9 for label 1 and ≤0.1 for label 0), the remaining dataset still contains several thousand KOIs in each class (≈3,600 label 0 and ≈2,900 label 1), so row-wise deletion for rare missing values does not harm statistical power. Moreover, it avoids introducing unphysical "average" values for quantities that span multiple orders of magnitude (e.g., radii, insolation, periods).
- Explain handling of extreme outliers (retain/remove/truncate) and provide brief rationale.

3.5 Exploratory Data Analysis (EDA)

- `koi_score` distribution (histogram/KDE): Show the bimodal distribution with peaks near 0 and 1, justifying the threshold-based labeling strategy.
- Visualize the final binary label distribution (class balance) after applying thresholds.
- Histograms and box plots of key features: general range and outlier distribution.
- Correlation matrix heatmap: showing multicollinearity among main features (e.g., whether period, duration, depth, SNR are highly correlated).
- Optional: Plot feature distributions or principal component scatter plots colored by binary label or `koi_disposition` to preview class separability.

---

4. Methodology

> Corresponds to the theoretical part of FRAMEWORK Step 2–3 + classification theory.

Suggested subsections:

4.1 Problem Formulation and Notation

- Mathematical formalization:
  - Given feature vector $\mathbf{x}_i \in \mathbb{R}^p$, the target is binary $y_i \in \{0, 1\}$ derived from `koi_score` via thresholding;
  - Objective: learn mapping $f : \mathbb{R}^p \to \{0, 1\}$, optimizing F1-score.
- Briefly state: Subsequently, we will train and compare classifiers in both original feature space $\mathbf{x}_i$ and PCA feature space $\mathbf{z}_i$.

4.2 Principal Component Analysis (PCA)

- Briefly describe PCA theory:
  - Standardization, covariance matrix, eigenvalue decomposition;
  - Principal components = linear combinations of original features, sorted by explained variance.
- Specify practical application settings:
  - **PCA is implemented using the Python `pca` library** (the dedicated PCA package);
  - The `pca` library is used to:
    - Compute principal components;
    - Generate scree plots, biplots, PC1–PC2 scatter plots, and loading plots;
  - All numerical PCA results and visualizations in this project are obtained from this library implementation;
  - Perform PCA on standardized numerical features (does not include target or label-leakage fields);
  - Use explained variance ratio with scree plot and cumulative curve to select the top k principal components to retain (typically reaching ~80–90% cumulative variance).
- Explain the purpose of PCA:
  - Dimensionality reduction;
  - Reduce multicollinearity;
  - Form interpretable "comprehensive physical directions" (e.g., geometry+SNR principal component, stellar scale principal component,

etc.);

- Used both for classification (input to PyCaret) and for interpretability (visualizations and loadings).

4.3 Classification Algorithms

- List three classification algorithms for detailed discussion and briefly explain their characteristics:

1. **Logistic Regression (LR)**
    - Linear probabilistic classifier
    - Uses logistic (sigmoid) function to model probability: $P(y = 1|\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}}}$
    - Strengths: Highly interpretable, efficient, works well when classes are approximately linearly separable
    - Weaknesses: Cannot capture non-linear decision boundaries
    - Connection to this project: Serves as a linear baseline for classifying high vs low quality KOIs; coefficient magnitudes indicate feature importance
2. **Random Forest (RF)**
    - Ensemble of decision trees, each trained on a bootstrap sample
    - Final prediction is made by majority vote across all trees
    - Strengths: Handles non-linear relationships, robust to outliers, provides feature importance rankings
    - Weaknesses: Less interpretable than linear models, can overfit with too many trees or insufficient pruning
    - Connection to this project: Can capture complex interactions between KOI features (e.g., SNR × stellar radius) that affect quality classification
3. **Multi-Layer Perceptron (MLP)**
    - Feedforward neural network with one or more hidden layers
    - Each layer applies: $\mathbf{h} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$ where σ is an activation function (e.g., ReLU)

    - Strengths: Universal function approximator, can learn highly non-linear decision boundaries
    - Weaknesses: Requires careful tuning (learning rate, architecture), prone to overfitting, less interpretable
    - Connection to this project: Can model complex non-linear patterns in KOI features that separate high-quality from low-quality candidates
- Emphasize: All three models (plus additional models) are trained and evaluated via **PyCaret's classification module**, not by custom per-model code. PyCaret systematically compares multiple classifiers on both feature spaces, and LR, RF, and MLP are selected for detailed discussion as representative algorithms.

---

5. Experimental Setup

> Consolidates implementation details; corresponds to FRAMEWORK descriptions of data splitting, standardization, PyCaret usage, and evaluation metrics.

Suggested subsections:

5.1 Train/Test Split and Scaling

- Explain split strategy (e.g., 70% / 30%):
    - Training set for model fitting via PyCaret;
    - Test set only for final evaluation.
- Standardization:
    - Use `StandardScaler`;
    - `fit` on training set, apply same transformation to train/test;
    - PCA and all classification models work in standardized space;
    - This produces `X_std` (standardized original features).

5.2 PyCaret Classification on Two Feature Spaces

- Describe the two experimental tracks:

1. **Classification on Original Standardized Features (`X_std`)**
    - Use `pycaret.classification` with `X_std` + binary label
    - Call `setup()` to initialize the classification environment
    - Call `compare_models()` to evaluate multiple classifiers
    - **Primary optimization metric: F1-score**
    - Secondary metrics: Accuracy, ROC-AUC
2. **Classification on PCA Features (`Z_pca`)**
    - Use the binary label with PCA scores as inputs
    - Run a separate `setup()` + `compare_models()` on PCA features
    - Use F1-score as primary metric for consistency
    - Record Accuracy and ROC-AUC

- Note: **PyCaret's internal `pca` option is not used**. We perform PCA explicitly using the `pca` library to maintain full control over the transformation and interpretation.

5.3 Model Selection and Focus Algorithms

- Explain that PyCaret trains and compares many classifiers automatically
- Three algorithms are selected for detailed analysis in the report:
    - **Logistic Regression (LR)**
    - **Random Forest (RF)**
    - **Multi-Layer Perceptron (MLP)**
- These are chosen as either:
    - Top-performing models from PyCaret's comparison, or
    - Representative models for deeper analysis (linear baseline, ensemble tree, neural network)

5.4 Evaluation Metrics

- Clearly list classification evaluation metrics:
    - **Primary metric: F1-score**
        - Harmonic mean of precision and recall: $F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

        - Particularly important for this task because it balances the ability to correctly identify high-quality KOIs (recall) with minimizing false positives (precision)
    - **Secondary metrics:**
        - **Accuracy:** Overall fraction of correct predictions
        - **ROC-AUC:** Area under the receiver operating characteristic curve, measures ability to discriminate between classes across all classification thresholds
    - For selected models (LR, RF, MLP), also present:
        - **Confusion matrix:** Shows true positives, true negatives, false positives, false negatives
        - **ROC curves:** Visual representation of true positive rate vs false positive rate
        - Optionally: **Precision** and **Recall** broken down by class

---

6. PCA Results and Interpretation

> Corresponds to the results section of FRAMEWORK Step 2.

Suggested subsections:

6.1 Explained Variance and Choice of k

- Show scree plot and cumulative explained variance curve (generated by the `pca` library).
- Point out:
    - Variance proportion explained by the first few principal components;
    - Final choice of k (e.g., "the first 5 principal components explain 89% of total variance, so k=5 is selected");
    - This creates the PCA feature space `Z_pca` used in subsequent classification.

6.2 Principal Component Loadings and Physical Meaning

- Provide summary of loading matrix (can use table/heatmap/bar chart), showing original features with large weights on each principal component.
- Attempt physical interpretation of the first few principal components:
  - Example: PC1 may represent a combination of "strong signal quality (high SNR, appropriate depth/duration)";
  - Example: PC2 may be related to "stellar properties (temperature, radius, surface gravity)";
  - PC3 might capture "orbital characteristics and irradiation" etc.
- Use the `pca` library's biplot and loading visualizations to support interpretation.

6.3 Visualization in PCA Space

- Plot scatter plot of KOIs in (PC1, PC2) plane:
  - Color by binary label (high-quality vs low-quality) or by `koi_disposition`;
  - Observe whether the two quality classes show structural separation in principal component space;
  - This visual separation (or lack thereof) gives intuition about whether PCA features will aid classification.
- Optional: Simple biplot with feature vector arrows to show how original features project onto PC1 and PC2.

---

7. Classification Results

> Replaces regression results; corresponds to FRAMEWORK Step 3 & 4.

Suggested subsections:

7.1 Model Comparison on Original Features (`X_std`)

- Provide table: test set performance of multiple models from PyCaret on `X_std`:
  - Columns include: Model, F1-score, Accuracy, ROC-AUC;
  - Highlight top-performing models;
  - Focus on LR, RF, and MLP as representative algorithms.

- Brief analysis:
  - Which models achieve the highest F1-score?
  - How do linear (LR) vs non-linear (RF, MLP) models compare?

7.2 Model Comparison on PCA Features (`Z_pca`)

- Similar to 7.1, provide another table: performance of models on `Z_pca`.
- Comparative analysis:
  - Performance changes of each model on Original vs PCA features;
  - Does PCA improve linear models (LR) by reducing collinearity?
  - Does PCA affect non-linear models (RF, MLP) differently?
  - Which feature space yields better overall F1-score?

7.3 Detailed Analysis of Focus Models (LR, RF, MLP)

- For each of the three focus algorithms, in both feature spaces:

  Confusion Matrices:

  - Show true positives, false positives, true negatives, false negatives
  - Analyze: Are false negatives (missing high-quality KOIs) or false positives (incorrectly labeling low-quality as high) more problematic?

  ROC Curves:

  - Plot ROC curves for LR, RF, MLP (on best-performing feature space or both)
  - Compare ROC-AUC values
  - Discuss trade-offs between true positive rate and false positive rate

  Precision and Recall:

  - Break down by class (especially positive class: high-quality KOIs)
  - Discuss: Does the model have high precision (few false alarms) and high recall (catches most true high-quality KOIs)?

7.4 Feature Importance and Model Interpretability

- For **Random Forest (on `X_std`)**:

    - List top feature importance rankings
    - Discuss which original features are most discriminative for quality classification
    - Example: SNR, depth, stellar radius might be highly important
- For **Logistic Regression (on `X_std` and/or `Z_pca`)**:
    - Report coefficient magnitudes
    - For `X_std`: interpret which features have strongest positive/negative effects on high-quality classification
    - For `Z_pca`: combine PC loadings with LR coefficients to trace back to original features
- Optional: SHAP values or permutation importance for deeper interpretability (brief mention if time permits)

---

8. Discussion

> Corresponds to the comprehensive comparison and interpretation section of FRAMEWORK Step 4.

Should include:

8.1 Summary of Model Comparison

- Synthesize results from Sections 7.1 and 7.2 to answer:
    - Which model + feature space combination achieves the **best F1-score**?
    - Which combination achieves the best ROC-AUC and Accuracy?
    - How does PCA impact classification performance?
        - Does it improve stability or interpretability for linear models?
        - Does it reduce performance for non-linear models due to information loss?
    - Trade-offs between different models:
        - Interpretability (LR > RF > MLP)
        - Performance (varies by feature space)
        - Training cost and complexity

8.2 Implications for Quality Assessment of KOIs

- Discuss from a quality engineering perspective:
  - How well can the classifier separate high-quality from low-quality KOIs?
  - Is the classifier reliable enough for practical pre-screening?
    - High recall means few high-quality candidates are missed
    - High precision means low false alarm rate
  - Can this model be used as a rapid quality screening tool?
    - Example workflow: Run classifier on new KOIs → prioritize high-scoring ones for expensive vetting pipeline → save computational resources
  - How does this lightweight classifier compare conceptually to the full Robovetter pipeline?

8.3 Impact of PCA on Classification

- Discuss specific findings about PCA:
  - Does PCA help linear models (LR) by reducing multicollinearity?
  - Does PCA hurt tree-based models (RF) by mixing features and reducing their ability to split on individual attributes?
  - Does PCA affect neural network (MLP) training or generalization?
  - Interpretability trade-off: PCA features are harder to interpret directly, but PC loadings can still provide insight

8.4 Limitations and Threats to Validity

- `koi_score` itself contains noise and may change with pipeline updates;
- Binary thresholding discards mid-range samples, which might contain valuable information;
- Unused features:
  - Uncertainty columns (`*_err1`, `*_err2`) not included;
  - Pixel-level data, time series information not used;
  - Sky coordinates (`ra`, `dec`) excluded, though spatial systematics might exist;

- Model may be affected by training sample distribution bias or class imbalance;
- PCA mixes features, which may weaken some astronomical interpretability for non-linear models.

---

9. Conclusion and Future Work

> Corresponds to "Conclusion: Report and Deliverables" in FRAMEWORK 1.3.

Should include:

9.1 Conclusion

- Review:
  - Motivation for selecting KOI dataset and binary quality classification task;
  - Overall workflow: EDA → binary label construction → PCA (via `pca` library) → PyCaret classification (focusing on LR, RF, MLP) → evaluation using F1-score, Accuracy, ROC-AUC;
- Summarize main findings:
  - Best-performing model and feature space combination (e.g., "Random Forest on original features achieves F1 = 0.XX");
  - Impact of PCA on different model types;
  - Most important features or principal components for quality classification;
  - Feasibility of using this classifier as a pre-screening tool for KOI quality assessment.

9.2 Future Work

- Possible extension directions:
  - **Include additional features:**
    - Uncertainty columns (`*_err1`, `*_err2`) for probabilistic modeling;
    - False positive flags as features (with careful cross-validation to avoid leakage);

        - Time series features from raw light curves;
    - **Try advanced models:**
        - Gradient Boosting (XGBoost, LightGBM);
        - Gaussian Process classifiers for uncertainty quantification;
        - Deep learning with time series inputs (LSTMs, 1D CNNs);
    - **Multi-class classification:**
        - Instead of binary high/low, classify into multiple quality tiers or specific false positive types;
    - **Extend to related tasks:**
        - Terrestrial planet detection (Earth-size candidates);
        - Habitable zone classification;
        - Direct prediction of planet vs false positive (instead of quality score);
    - **Improve interpretability:**
        - SHAP analysis for black-box models;
        - Counterfactual explanations: "What would it take to flip this low-quality KOI to high-quality?";
    - **Bayesian methods:**
        - Probabilistic classifiers with uncertainty estimates;
        - Better handling of noisy labels and class overlap.

---

10. References

- Papers & documentation:
    - Kepler mission, Robovetter, `koi_score` official documentation and papers;
    - NASA Exoplanet Archive documentation;
    - PCA textbooks or papers;
    - Classification algorithms (Logistic Regression, Random Forest, MLP) papers/books;
    - PyCaret documentation and papers;

    - Python `pca` library documentation;
    - Evaluation metrics (F1-score, ROC-AUC) references;
    - If interpretability analysis (SHAP, permutation importance, etc.) is included, cite corresponding papers/books.
- Organize according to IEEE or your chosen citation format.