FRAMEWORK

1. High-level Instruction (Motivation → Process → Conclusion)

1.1 Motivation: Why Choose the KOI Dataset and `koi_score`

Task Background

The Kepler mission produced a large number of KOIs (Kepler Objects of Interest), where each KOI represents a signal that "looks like a planetary transit."

These signals include:

- Actual planet candidates
- Various false positives (binary stars, background stars, noise, etc.)

The Role of `koi_score`

The Kepler team uses an automated vetting system (**Robovetter**) to classify each KOI as:

- Candidate / False Positive
- And assigns a `koi_score` **between 0–1**, indicating how "reliable" the classification is

💡 **What is `koi_score`?**
A **quality/confidence metric** that integrates numerous tests and Monte Carlo experiments. In subsequent planet statistics and occurrence rate studies, it is often used with a threshold (e.g., ≥0.9) to select "high-quality samples."

Why Choose This Dataset and Target

Dataset Advantages:

- The KOI cumulative table is a **public, real-world, moderately-sized, multi-variable engineering dataset**
- Ideally suited for the course requirements of PCA + ML

Target Significance:

- `koi_score` is not a simple physical quantity, but rather a "**quality score output from a complex pipeline**," which closely aligns with the *Advanced Statistical Approaches to Quality* theme
- Computing it requires the full Robovetter suite plus extensive injection tests, making it non-reproducible and costly for ordinary users

🎯 **Core Question of This Project**

> Can we use only the physical and observational features available in the KOI table, and apply statistical & machine learning methods to **approximate and reconstruct** **`koi_score`**, creating a lightweight "quality assessment surrogate model"?

1.2 Process: EDA + PCA → Four Regression Model Comparison

Target Problem (High Level)

> Given the multi-dimensional features of KOIs (orbital period, transit depth, duration, SNR, stellar parameters, etc.), first perform standardization and exploratory data analysis (EDA), then use PCA to extract key principal components. Train four regression models—**Linear Regression, Ridge Regression, RandomForestRegressor, MLPRegressor**—on both the original feature space and the PCA feature space to predict the continuous `koi_score`, and compare which combination is best suited as a "lightweight quality assessor."

The overall workflow draws from the sample project: first perform **data cleaning + standardization + PCA visualization**, then conduct **multi-model systematic comparison**, and finally interpret and discuss the results.

Step 1 – Data Acquisition, Cleaning, and Exploratory Data Analysis (EDA)

Step 1.1 Data Acquisition and Field Selection (KOI Columns to Retain)

Download the KOI table from NASA Exoplanet Archive (with complete header). After reading it in, **don't throw everything directly into the model**, but rather categorize fields into 4 functional groups:

(a) Identifier Fields (For Indexing Only, Not Features)

These columns are only used to label each KOI record and **do not enter the regression model**:

- `kepid` – Kepler Catalog ID
- `kepoi_name` – KOI Name
- `kepler_name` – Official Kepler Planet Name (if any)

These fields can be used to trace back to specific KOI individuals in the original table during subsequent analysis/plotting.

(b) Target Variable (Label)

`koi_score` – Disposition Score

- This is the project's only regression target y, with values in [0,1], representing vetting classification confidence
- Problem definition: Given KOI physical and observational features, predict the continuous `koi_score`

(c) Labels and False Positive Flags for Analysis/Grouping Only (Not in Feature Matrix X)

These fields are used for "slicing/comparing" in EDA and results discussion, but **not as model predictors** to avoid label leakage:

Classification Result Types:

- `koi_disposition` – Exoplanet Archive Disposition
- `koi_pdisposition` – Disposition Using Kepler Data

False Positive Flags:

- `koi_fpflag_nt` – Not Transit-Like False Positive Flag
- `koi_fpflag_ss` – Stellar Eclipse False Positive Flag
- `koi_fpflag_co` – Centroid Offset False Positive Flag
- `koi_fpflag_ec` – Ephemeris Match Indicates Contamination False Positive Flag

⚠️ These fields reflect the Robovetter/pipeline's own judgment results or intermediate states. If used to predict `koi_score`, they would highly overlap with the target. Therefore, they are only used in **description and visualization**, not included in the PCA or the four regression models' feature vectors.

(d) Primary Feature Columns for Models (Core Numerical Features of X)

The following are the **primary independent variables** for this project's PCA and four regression models, all from the "central value" columns in the header:

1. Transit Geometry & Shape

- `koi_period` – Orbital Period [days]
- `koi_time0bk` – Transit Epoch [BKJD]
- `koi_impact` – Impact Parameter
- `koi_duration` – Transit Duration [hrs]
- `koi_depth` – Transit Depth [ppm]
- `koi_model_snr` – Transit Signal-to-Noise

2. Planet Properties & Irradiation

- `koi_prad` – Planetary Radius [Earth radii]
- `koi_teq` – Equilibrium Temperature [K]
- `koi_insol` – Insolation Flux [Earth flux]

3. Stellar Properties

- `koi_steff` – Stellar Effective Temperature [K]
- `koi_slogg` – Stellar Surface Gravity [log10(cm/s^2)]
- `koi_srad` – Stellar Radius [Solar radii]

4. Sky Position & Brightness

- `ra` – RA [decimal degrees]
- `dec` – Dec [decimal degrees]
- `koi_kepmag` – Kepler-band [mag]

5. Pipeline Metadata Not Entering Model (Drop from X)

- `koi_tce_plnt_num` – TCE Planet Number (just the numbering on the same star, lacks clear physical meaning)
- `koi_tce_delivname` – TCE Delivery (processing batch name, data versioning information)

(e) Strategy for Uncertainty Columns (_*err1, _*err2) in This Project

For upper/lower bound uncertainty columns appearing in the header, such as:

`koi_period_err1`, `koi_period_err2`, `koi_duration_err1`, `koi_duration_err2`, `koi_depth_err1`, `koi_depth_err2`, `koi_prad_err1`, `koi_prad_err2`, `koi_teq_err1`, `koi_teq_err2`, `koi_insol_err1`, `koi_insol_err2`, `koi_steff_err1`, `koi_steff_err2`, `koi_slogg_err1`, `koi_slogg_err2`, `koi_srad_err1`, `koi_srad_err2`

Processing Strategy:

- **Not retained**
- In the main experiment ("PCA + four regression models"), to avoid dimension explosion and further aggravating collinearity, the **baseline only uses central value columns as features**

Step 1.2 Basic Data Inspection and EDA

Basic Data Inspection (Following Sample's seeds.ipynb)

Use `df.info()`, `df.describe()`, `df.duplicated().sum()` to check:

- Row count, column count, data types
- Whether duplicate rows exist
- Whether any variables have severe missingness

Visualization:

- Histogram / KDE of `koi_score` to observe whether its distribution is skewed
- Boxplots / violin plots of numerical features to check for outliers

- Correlation matrix heatmap to examine multi-collinearity among features

Feature Selection and Label Leakage Control:

- Independent variables X: Only use observational/physical features, not classification results themselves
- Dependent variable y: `koi_score`
- "Classification label" fields like `koi_disposition` / `koi_pdisposition`: **Not used as model input features** to avoid label leakage; only used as auxiliary variables in EDA and results discussion

Missing Values and Outlier Handling:

- Remove samples with missing `koi_score`
- Use mean/median imputation for features with few missing values
- Reasonably handle extreme outliers (delete or winsorize as appropriate, but avoid over-cleaning)

Data Splitting and Standardization (Corresponding to StandardScaler in Sample):

Divide data into train / validation / test sets:

- Training set 70% (for model fitting and cross-validation)
- Validation set 15% (for model selection / hyperparameter tuning)
- Test set 15% (only for final evaluation)

Use `StandardScaler` for all numerical features:

- `fit` on the training set, then `transform` train/val/test with the same scaler
- Ensure PCA and subsequent models all work in the standardized space

---

Step 2 – PCA (Principal Component Analysis) and Principal Component Visualization

Perform PCA on Standardized Features

Use `sklearn.decomposition.PCA` and/or the `pca` third-party library:

- First fit all principal components without limiting the number
- Calculate the **explained variance ratio** for each principal component

Determine Number of Principal Components to Retain

Plot explained variance and cumulative explained variance curves (similar to sample):

- Look for an "elbow" around 90%–95% cumulative variance to determine appropriate `k`
- Select the first `k` principal components as the PCA feature space, denoted as `PC1, PC2, ..., PCk`

Principal Component Space Visualization (Referencing Sample's PCA Plots)

Scatter Plot Visualization:

- Plot all KOIs on the (PC1, PC2) plane, using `koi_disposition` as color labels
- Intuitively observe: whether true planet candidates and false positives show structural differences in the principal component space

Feature Projection Visualization (Loadings/Biplot):

- Show which original features have large weights on PC1, PC2
- Interpret from a physical perspective what combination each principal component roughly represents (e.g., "geometry/SNR component," "stellar scale/temperature component," etc.)

Generate PCA Feature Dataset

Obtain two versions of feature matrices:

- Original standardized features: `X_std`
- PCA principal component features: `X_pca` (retaining only the first `k` principal components)

The subsequent four regression models will be trained and compared on these two feature sets.

Step 3 – Four Regression Models on Original Features (Baseline)

Train and compare the following four regression models on the standardized original features `X_std`:

1. Linear Regression (Ordinary Least Squares Linear Regression)

- Use `sklearn.linear_model.LinearRegression`
- Serves as the most basic linear baseline model: no regularization; convenient for comparison with Ridge Regression

2. Ridge Regression (Linear Regression with L2 Regularization)

- Use `sklearn.linear_model.Ridge`
- Select regularization strength `alpha` through cross-validation (e.g., using `RidgeCV` or GridSearchCV)
- **Objective:** Test whether regularization can improve generalization performance and reduce variance in the presence of multi-collinearity and noise

3. RandomForestRegressor (Random Forest Regression)

- Use `sklearn.ensemble.RandomForestRegressor`
- Main hyperparameters: `n_estimators`, `max_depth`, `min_samples_leaf`, etc.
- Find appropriate parameter configurations through cross-validation, recording:
  - Prediction performance (R^2, RMSE, MAE)
  - Model feature importance

4. MLPRegressor (Multi-layer Perceptron Regression)

- Use `sklearn.neural_network.MLPRegressor`
- Design a simple but reasonable network structure (e.g., 1–2 hidden layers, ReLU activation), and search the following parameters:
  - `hidden_layer_sizes`, `alpha`, `learning_rate_init`, `max_iter`, etc.
- Focus on MLP's ability to fit non-linear relationships and its sensitivity to data scale and regularization

Evaluation and Comparison

- Use K-fold cross-validation on the training set to evaluate each model, recording:

    - Cross-validation R^2, RMSE, MAE mean and standard deviation
    - Training time, model complexity, stability
- Evaluate final model performance on the independent test set, forming a baseline results table

---

Step 4 – Four Regression Models on PCA Principal Components (PCA + Regression)

On PCA features `X_pca` (the first `k` principal components), use the exact same four regression models, replicating Step 3's training/tuning/evaluation workflow.

Model Structure Remains Consistent

Still use:

- Linear Regression
- Ridge Regression
- RandomForestRegressor
- MLPRegressor

But input features switch from original standardized features to PCA principal components.

Training and Validation

- Use the same data split scheme (train/val/test) to ensure comparability
- Conduct the same cross-validation and hyperparameter search procedures
- Record the same metrics: R^2, RMSE, MAE, and training time

Analysis of PCA's Impact on Different Models

Compare the performance changes of each model under "original features vs PCA features":

- Whether linear models become more stable due to PCA alleviating collinearity
- Whether Random Forest degrades after PCA (because tree models are more sensitive to local partitioning of original features)

- Whether MLP converges more easily after PCA, and whether overfitting is mitigated

---

Step 5 – Comprehensive Comparison and Interpretation

Results Summary and Visualization

Organize all experimental results into tables and figures:

- **Rows:** 4 models (Linear / Ridge / RF / MLP)
- **Columns:** Original features vs PCA features, R^2, RMSE, MAE, training time, etc.

Draw comparison plots (e.g., bar charts / error bar plots) to help intuitively compare the performance of different models and feature spaces.

Discussion of Best Model and PCA Value

Answer the following questions:

- Which model performs best overall in predicting `koi_score`?
- Is PCA's impact on overall performance and model stability positive or negative?
- Considering actual deployment costs (training time, model complexity), which combination is most suitable?

Interpretation of Principal Components and Original Features

Utilize:

- Random Forest feature importance (without PCA)
- PCA loadings (with PCA)
- If necessary, use SHAP / permutation importance for interpretability analysis of the best model

Discuss:

- Which original features or principal components are most sensitive to `koi_score`

- Their rough astrophysical meanings (e.g., "high SNR + reasonable geometry" corresponds to high `koi_score`, etc.)

Return to Task Motivation (Echoing Section 1.1)

Summarize the project's conclusions:

- Using EDA + PCA + four regression models, what prediction accuracy can be achieved for KOI's `koi_score`
- Can these models serve as a "lightweight, interpretable" auxiliary tool for NASA's original vetting pipeline
- What limitations exist and future improvement directions (more features, time series, Bayesian models, etc.)

1.3 Conclusion: Report and Deliverables

Final Deliverables Include:

1. Jupyter Notebook (on GitHub)

- Complete code from data loading, preprocessing, PCA, model training to results visualization
- Clear markdown explanations for easy reproduction by the instructor

2. 7–10 Page Two-Column PDF Report

- Use IEEE two-column format or similar
- Report cover page includes:
  - Your name
  - Student ID
  - GitHub repository link

3. Complete Project Narrative in Report

- Why choose KOI & `koi_score` (Motivation)
- How to design PCA + Linear Regression + MLP pipeline (Process)
- Model performance and significance for "quality assessment/reliability" (Conclusion)