A Framework for Machine Learning-Driven Exoplanet Discovery and Analysis: From Raw Data to Interactive Web Platform

The Scientific Foundation of Transit Photometry and Exoplanet Datasets

The quest to discover planets beyond our solar system, known as exoplanets, has transformed from a speculative endeavor into a data-driven science. Central to this transformation is the transit photometry method, an indirect detection technique that has proven to be the most successful strategy to date, responsible for the discovery of thousands of exoplanets.[1] This method, employed by groundbreaking space telescopes like NASA's Kepler, K2, and the Transiting Exoplanet Survey Satellite (TESS), has generated vast archives of observational data. The sheer volume and complexity of this data necessitate the use of automated analysis pipelines, where machine learning (ML) has emerged as a powerful tool for classifying potential exoplanet candidates. This report outlines a comprehensive framework for developing an end-to-end exoplanet detection system, beginning with the foundational principles of transit photometry and the characteristics of the datasets it produces, and culminating in the architectural design of a sophisticated ML model and an interactive web platform for scientific analysis.

The Transit Method: Detecting Shadows Across the Cosmos

The transit method is predicated on a simple yet elegant principle: observing the minute, periodic dimming of a star's light as an orbiting planet passes in front of it from our line of sight.[3] This event, known as a transit, causes a temporary dip in the star's measured brightness. By continuously monitoring a star's flux over time, astronomers can generate a "light curve"—a graph of brightness versus time. A repeating, U-shaped or "bucket-shaped" dip in this light curve is the characteristic signature of a transiting exoplanet.[4]

The power of the transit method lies in the wealth of information that can be extracted from the light curve. The periodicity of the transits directly corresponds to the planet's orbital period, which, through Kepler's laws of planetary motion, allows for the calculation of its orbital size.[3] The depth of the transit—how much the starlight dims—is proportional to the ratio of the planet's area to the star's area. Since the star's size can often be estimated with high accuracy from its spectral type and other properties, the transit depth provides a direct measurement of the planet's radius.[4] This ability to determine a planet's physical size is a unique advantage of the transit method over other indirect techniques like the radial velocity method, which primarily constrains a planet's mass. When both methods can be applied to the same system, the combination of mass and radius yields the planet's bulk density, a crucial clue to its composition—whether it is a rocky terrestrial world, a gaseous giant, or something in between.[4]

Despite its success, the transit method has inherent observational biases. The probability of a planet's orbit being aligned edge-on to our line of sight is geometrically small, and this probability decreases for planets with larger orbits. Consequently, the method is most effective at discovering planets with short orbital periods that are close to their host stars, as they transit more frequently and have a higher geometric probability of being detected.[7] Furthermore, the transit signal is more pronounced for larger planets orbiting smaller stars. An Earth-sized planet transiting a Sun-like star produces a minuscule dimming of about 84 parts per million (ppm), whereas the same planet orbiting a smaller, cooler M-dwarf star would cause a much more significant and easily detectable drop in brightness.[1] This creates a bias towards finding large planets, such as "hot Jupiters," and planets orbiting smaller stars.[7]

A primary challenge in transit-based surveys is distinguishing genuine planetary signals from "false positives." These are astrophysical or instrumental phenomena that mimic the signature of a transiting planet. The most common source of false positives is an eclipsing binary (EB) star system, where two stars orbit each other. If one star is much smaller and dimmer than the other, its eclipse can produce a light curve dip similar to a planetary transit. However, these events often produce a more V-shaped light curve compared to the flat-bottomed, U-shape of a planetary transit, a distinction that is critical for classification.[5] Other false positives can arise from background eclipsing binaries contaminating the photometric aperture of the target star or from instrumental artifacts. The rigorous process of vetting these false positives is a central task for any exoplanet detection pipeline.

A Comparative Analysis of NASA's Exoplanet Datasets

This project utilizes three seminal, publicly available datasets from NASA's exoplanet missions: the Kepler Objects of Interest (KOI) cumulative table [8], the K2 Planets and Candidates table [8], and the TESS Objects of Interest (TOI) table.[8] It is crucial to recognize that these datasets are not the raw photometric time-series data (i.e., light curves or FITS files). Instead, they are high-level, pre-processed scientific products. Each row represents a "Threshold Crossing Event" (TCE)—a periodic signal detected by the mission's data processing pipeline—and the columns contain derived physical parameters from fitting a transit model to the underlying light curve. This distinction is fundamental; our initial task is not to perform signal processing on raw light curves but to build a classification model based on these derived, tabular features.

A significant challenge in utilizing these datasets together is their heterogeneity. Each mission's science team used slightly different data processing pipelines and naming conventions, resulting in disparate schemas. To build a single, robust model, these datasets must be harmonized into a unified structure.

Data Harmonization: The first step is to map the different column names to a consistent, standardized schema. The target classification label, for instance, is named koi_disposition in the Kepler data [8],

disposition in the K2 data [8], and

tfopwg_disp (TESS Follow-up Observing Program Working Group disposition) in the TESS data.[8] Similarly, physical parameters like orbital period or planetary radius have different prefixes (e.g.,

koi_period vs. pl_orbper). A clear mapping is required to merge these sources.

Target Variable Definition: The ultimate goal is to classify each object of interest into one of three categories. We will standardize the labels from all datasets into a single target variable with the following classes:

- **CONFIRMED:** The object has been rigorously vetted and confirmed as a genuine exoplanet.
- **CANDIDATE:** The signal is consistent with a planetary transit but has not yet been fully validated. These are high-priority targets for follow-up observation. In the Kepler and K2 data, this corresponds to the CANDIDATE label.[8] In the TESS data, this corresponds to PC (Planet Candidate).[8]
- **FALSE POSITIVE:** The signal has been determined to originate from a non-planetary source, such as an eclipsing binary or instrumental artifact. This corresponds to the FALSE POSITIVE label in Kepler/K2 and FP in TESS.[8]

Key Feature Identification: An initial survey of the datasets reveals a rich set of features that are critical for classification. These can be grouped into several categories [8]:

- **Orbital Parameters:** These describe the orbit of the candidate, such as pl_orbper (Orbital Period) and pl_tranmid (Transit Midpoint Time). The periodicity of the signal is a fundamental characteristic of a true planet.
- **Transit Signal Parameters:** These quantify the shape and significance of the light curve dip, including pl_trandurh (Transit Duration), pl_trandep (Transit Depth), and koi_impact (Impact Parameter, which describes how centrally the planet crosses the star).
- **Derived Planetary/Candidate Parameters:** These are physical properties of the object itself, inferred from the transit signal and stellar properties. Key examples include pl_rade (Planetary Radius), pl_eqt (Equilibrium Temperature), and pl_insol (Insolation Flux).
- **Stellar Parameters:** These describe the host star, which is essential for correctly interpreting the transit signal. They include st_teff (Stellar Effective Temperature), st_rad (Stellar Radius), and st_logg (Stellar Surface Gravity).
- **False Positive Flags:** The Kepler dataset, in particular, contains a set of highly informative binary flags (koi_fpflag_nt, koi_fpflag_ss, koi_fpflag_co, koi_fpflag_ec).[8] These flags are the output of NASA's own automated vetting pipeline, designed to flag signals that are not transit-like, resemble a stellar eclipse, show a significant centroid offset (indicating a background source), or have an ephemeris match with a known contaminating source.[9] These columns are not raw measurements but a form of encoded expert knowledge, making them exceptionally powerful features for a machine learning model.

The different missions also carry distinct observational biases. The original Kepler mission stared at a single patch of sky for four years, allowing it to detect planets with longer orbital periods.[3] In contrast, TESS surveys nearly the entire sky, but observes each sector for only about 27 days, making it more sensitive to short-period planets around bright, nearby stars.[10] The K2 mission was a re-purposing of the Kepler satellite after a mechanical failure and had its own unique instrumental challenges.[12] A successful model trained on a combined dataset must be robust enough to generalize across these different sources, which underscores the importance of careful feature selection and normalization. The table below provides a concrete blueprint for harmonizing these disparate datasets into a single, model-ready format.

Unified Feature Name	Kepler KOI Column [8]	K2 Column [8]	TESS TOI Column [8]	Description
target_disposition	koi_disposition	disposition	tfopwg_disp	The final classification label (Target

				Variable)
fp_flag_nt	koi_fpflag_nt	N/A	N/A	Not Transit-Like Flag
fp_flag_ss	koi_fpflag_ss	N/A	N/A	Stellar Eclipse / Secondary Eclipse Flag
fp_flag_co	koi_fpflag_co	N/A	N/A	Centroid Offset Flag
fp_flag_ec	koi_fpflag_ec	N/A	N/A	Ephemeris Match Flag
orbital_period_days	koi_period	pl_orbper	pl_orbper	Orbital period of the candidate in days
transit_epoch_bjd	koi_time0bk	pl_tranmid	pl_tranmid	Midpoint time of the first observed transit (BJD)
transit_duration_hr	koi_duration	pl_trandur	pl_trandurh	Duration of the transit in hours
transit_depth_ppm	koi_depth	pl_trandep	pl_trandep	Depth of the transit in parts per million (ppm)
planet_radius_earth	koi_prad	pl_rade	pl_rade	Planetary radius in units of Earth radii
equilibrium_te	koi_teq	pl_eqt	pl_eqt	Equilibrium

mp_k				temperature of the planet in Kelvin
insolation_flux_earth	koi_insol	pl_insol	pl_insol	Insolation flux relative to Earth
impact_parameter	koi_impact	pl_imppar	N/A	Transit impact parameter
stellar_eff_temp_k	koi_steff	st_teff	st_teff	Effective temperature of the host star in Kelvin
stellar_logg_cm_s2	koi_slogg	st_logg	st_logg	Stellar surface gravity (log10(g) in cm/s^2)
stellar_radius_solar	koi_srad	st_rad	st_rad	Stellar radius in units of Solar radii

Table 1: Data Harmonization and Feature Mapping. This table provides a clear schema for merging the Kepler, K2, and TESS datasets. It defines a unified feature name for each key parameter and maps it to the corresponding column in each of the source files. This serves as the blueprint for the initial data ingestion and cleaning script, ensuring consistency across the combined dataset. Note that some features, like the detailed false positive flags, are unique to the Kepler dataset and will result in null values for K2 and TESS entries, a factor that must be addressed during imputation.

The Data Processing Pipeline: From Noisy Observations to Model-Ready Features

The raw, merged dataset, while comprehensive, is not immediately suitable for training a high-performance machine learning model. It contains missing values, potential outliers, and a feature set that, while informative, can be significantly enhanced through domain-specific engineering. This section details a rigorous data processing pipeline designed to transform the initial tabular data into a clean, feature-rich, and balanced dataset optimized for classification. Each step is critical for ensuring the model's robustness, accuracy, and scientific validity.

Data Cleaning and Imputation

The first stage of the pipeline involves merging the disparate datasets and addressing data quality issues, particularly missing values.

Merging Datasets: Using the harmonization schema defined in Table 1, the three datasets—Kepler KOI, K2, and TESS TOI—are loaded into a single pandas DataFrame. This process involves renaming columns to match the unified schema and standardizing the categorical labels of the target_disposition column.

Handling Missing Values: Astronomical datasets frequently contain missing values due to a variety of reasons, including instrument limitations, data corruption, or the simple fact that not all parameters can be measured for every object.[13] A naive approach, such as dropping all rows with any missing data, would lead to a significant loss of valuable information. A more sophisticated strategy is required.

First, the pattern of missingness is analyzed. A visualization using a library like missingno can reveal which features are most affected and whether the missingness is correlated between columns.[14] For features with a very high percentage of missing values (e.g., > 50%), it may be prudent to discard the feature entirely, as imputation would be unreliable.

For features with a manageable number of missing values, imputation is performed. Simple statistical imputation, such as replacing missing values with the mean or median of the column, is generally discouraged for scientific data as it can distort the underlying data distribution and ignore correlations between features.[15] Instead, a model-based imputation strategy is recommended. The

IterativeImputer from scikit-learn is an excellent choice. This technique models each feature with missing values as a function of other features and uses that estimate for imputation. It iteratively cycles through the features, predicting each one based on the others, until the estimates converge. This approach preserves the relationships and correlations within the data, leading to more realistic and physically plausible imputations than simpler methods.[16] This choice reflects a key principle: the imputation method itself encodes assumptions about the data. Using a correlational model like

IterativeImputer assumes that a missing physical parameter (e.g., planetary radius) can be reasonably estimated from other measured parameters (e.g., transit depth, stellar radius), which is a far more robust assumption than assuming it is equal to the dataset's average.

Outlier Detection and Treatment

Outliers in astronomical data are a double-edged sword: they can represent instrumental errors or data processing artifacts that will degrade model performance, but they can also represent rare and scientifically interesting astrophysical phenomena.[17] Therefore, a blanket removal of all outliers is unwise. The recommended approach is to use an unsupervised anomaly detection algorithm to flag potential outliers for further review or careful treatment.

The **Isolation Forest** algorithm is particularly well-suited for this task. It works by randomly partitioning the data, isolating observations. The logic is that anomalous data points are "few and different," making them easier to isolate than normal points. The algorithm assigns an anomaly score to each data point based on how many partitions are required to isolate it. This method is computationally efficient and does not assume a specific distribution for the data, making it robust for complex, high-dimensional datasets.[18]

Once potential outliers are flagged, a decision must be made. For this project, a conservative approach is recommended: instead of removing the outliers, which could discard a novel type of exoplanet, their values for the most extreme features can be "capped" or "floored" at a reasonable percentile (e.g., the 1st and 99th percentiles). This process, known as winsorization, mitigates the influence of extreme values on the model's training without completely discarding the observation.

Advanced Feature Engineering: Creating Predictive Power

Feature engineering is arguably the most critical step for maximizing model performance. It involves creating new, synthetic features from the existing data to provide the model with more explicit and powerful predictive signals. This process is an act of embedding physical intuition and domain knowledge directly into the dataset, making the model's learning task easier and more effective.

Physical Ratios and Derived Parameters: Many of the most important relationships in exoplanet science are expressed as ratios or combinations of fundamental parameters.

- **Planetary Density:** Where planetary mass estimates are available (often from follow-up radial velocity measurements, present for some objects in the datasets), a density feature can be calculated as $ \rho_{p} \propto \frac{M_{p}}{R_{p}^{3}} $, where is the planet mass and is the planet radius. Density is a powerful discriminator between rocky planets, gas giants, and stellar objects.
- **Radius Ratio:** The transit depth, , is physically related to the square of the ratio of the planet's radius to the star's radius: $ \delta \approx (\frac{R_{p}}{R_{\star}})^{2} $.[6] Creating a feature for the radius ratio, planet_radius_earth / stellar_radius_solar, explicitly provides the model with this fundamental relationship.
- **Stellar Density:** The shape and duration of a transit are also related to the density of the host star. Stellar density can be estimated from the stellar mass and radius: $ \rho_{\star} \propto \frac{M_{\star}}{R_{\star}^{3}} $.

Signal-to-Noise Ratio (SNR) Features: The datasets provide error estimates for many parameters (e.g., pl_orbpererr1, koi_depth_err1). The reliability of a measurement is often more important than the measurement itself. A feature representing the signal-to-noise ratio can be created for each of these parameters by dividing the parameter's value by its associated error (e.g., SNR_period = koi_period / koi_period_err1). A high SNR indicates a confident, precise measurement and is a strong indicator of a genuine, well-defined signal rather than random noise.[19]

Frequency Domain Features: While this project begins with tabular data, it is worth noting a powerful technique for raw light curves. The **Fast Fourier Transform (FFT)** is a mathematical tool that decomposes a time-series signal into its constituent frequencies. For a light curve, this can reveal the strongest periodic signals present in the data. The period, amplitude, and phase of the most significant peaks in the frequency spectrum can be extracted and used as highly predictive features for any model, as they directly capture the periodic nature of a potential transit.[20] This technique would be central to any future extension of the model that processes raw FITS files.

Feature Scaling and Balancing

The final preprocessing steps ensure that the data is in an optimal format for the machine learning algorithm and that the model is not biased by the data's inherent structure.

Feature Scaling: Many machine learning algorithms, particularly those that rely on distance calculations like SVMs or regularization like Logistic Regression, are sensitive to the scale of the input features. A feature with a large range (e.g., insolation flux) could dominate the learning process over a feature with a small range (e.g., impact parameter). To prevent this, all numerical features must be scaled. **Z-score normalization (StandardScaler)** is the recommended method. It transforms each feature to have a mean of 0 and a standard deviation of 1, placing all features on a common scale without distorting the shape of their distributions. This is a standard and robust technique in exoplanet classification pipelines.[22]

Handling Class Imbalance: Exoplanet datasets are notoriously imbalanced. The number of "FALSE POSITIVE" objects vastly outnumbers the "CONFIRMED" exoplanets. If a model is trained on such an imbalanced dataset, it will likely develop a strong bias towards the majority class, achieving high accuracy simply by predicting "FALSE POSITIVE" for most inputs. This would result in a model with very poor recall for the scientifically valuable "CONFIRMED" class.

To counteract this, the **Synthetic Minority Over-sampling Technique (SMOTE)** is employed.[23] SMOTE works by creating new, synthetic samples of the minority class (in this case,

CONFIRMED and CANDIDATE). It selects a minority class instance, finds its k-nearest neighbors, and creates a new synthetic instance at a randomly selected point along the line segment connecting the instance and its neighbors. This process does not simply duplicate existing data; it populates the feature space around the minority class examples, effectively "thickening" the decision boundary the model needs to learn.[24] This forces the model to learn a more robust and generalized representation of what constitutes a confirmed planet, significantly improving its recall and overall performance.

Model Selection and Architecture Design

With a clean, feature-rich dataset prepared, the next critical step is to select and design the machine learning architecture. This involves a comparative analysis of suitable algorithms, a justified recommendation for the primary model, and a clear definition of the metrics that will be used to evaluate its success. The choice of model is not arbitrary; it is guided by the structure of the data, the specific classification task, and the need for scientific interpretability.

A Comparative Analysis of Candidate Models

The literature on machine learning for exoplanet detection highlights several powerful algorithms, each with distinct strengths and weaknesses. The primary candidates for this project are Random Forests, Support Vector Machines, and Convolutional Neural Networks.

Random Forest (RF): This is an ensemble learning method that operates by constructing a multitude of decision trees at training time. For a classification task, the final prediction is the mode of the classes output by individual trees. RF models are consistently praised in the exoplanet literature for their high accuracy and robustness.[25] Their key advantages include:

- **High Performance:** Studies report accuracies often exceeding 95% and sometimes reaching as high as 99.8% on Kepler data.[25]
- **Robustness to Feature Scaling:** As a tree-based method, RF is not sensitive to the scale of the input features, simplifying the preprocessing pipeline.
- **Implicit Feature Selection:** It can handle a large number of features and implicitly determines which ones are most important.
- **Interpretability:** RF models provide a direct measure of "feature importance," which is invaluable for scientific interpretation, allowing researchers to understand which physical parameters are most predictive.[22]
- **Low Overfitting Risk:** The ensemble nature of combining many trees (bagging) makes RF less prone to overfitting compared to a single decision tree.

Support Vector Machines (SVM): An SVM is a powerful classifier that works by finding an optimal hyperplane that best separates the different classes in the feature space. For non-linearly separable data, it uses the "kernel trick" to map the data into a higher-dimensional space where a linear separation is possible.

- **Strong Performance:** SVMs often achieve performance metrics very close to those of Random Forests, with reported accuracies around 93.8%.[27]
- **Effective in High Dimensions:** SVMs are effective in high-dimensional spaces, making them suitable for datasets with many features.
- **Sensitivity to Hyperparameters:** Their performance is highly dependent on the choice of the kernel (e.g., Radial Basis Function - RBF) and the tuning of hyperparameters like (regularization) and (kernel coefficient), often requiring careful optimization via grid search.[30]
- **Less Interpretable:** Compared to Random Forests, SVMs are more of a "black box." While techniques exist to estimate feature importance, they are not as direct or intuitive.

Convolutional Neural Networks (CNNs): CNNs are a class of deep learning models that have revolutionized image and sequence analysis. For exoplanet detection, they are typically applied directly to the raw or phase-folded light curves, which are treated as 1D time-series data or converted into 2D image-like representations (e.g., recurrence plots).[21]

- **State-of-the-Art on Raw Data:** When applied to light curves, CNNs excel at automatically learning hierarchical features—from simple dips to complex transit shapes and noise patterns—without the need for manual feature engineering.[32]

- **Data Representation Mismatch:** For the current project, which starts with pre-processed, tabular data, a standard CNN is not the appropriate tool. A simple Multi-Layer Perceptron (MLP) could be used, but ensemble methods like Random Forest typically outperform them on structured, tabular data.
- **High Complexity:** CNNs require significantly more data and computational resources to train effectively and are generally more complex to implement and tune than RF or SVM models.

This comparative analysis reveals a crucial architectural principle: the optimal model is dependent on the representation of the data. For the pre-extracted, tabular features available in the provided NASA datasets, ensemble methods like Random Forest are the superior choice. If the project were to evolve to process the raw FITS light curves directly, a CNN would become the state-of-the-art option.

Recommended Model: The Random Forest Classifier

Based on the evidence from the literature and the nature of the input data, the **Random Forest Classifier** is the recommended primary model for this project. Its consistent high performance on similar exoplanet classification tasks, its robustness, and, most importantly, its inherent interpretability make it the ideal choice for a tool intended for scientific analysis.[25] A scientist using the final web platform will not only receive a prediction but will also be able to understand which physical parameters drove that prediction, fostering trust and enabling deeper scientific insight.

For implementation, the RandomForestClassifier within Python's scikit-learn library is the industry standard. It is a robust, well-documented, and computationally efficient implementation that integrates seamlessly with the rest of the proposed data science stack.[35]

Model	Typical Accuracy (Exoplanet Data)	Required Data Representation	Computational Cost	Interpretability
Random Forest (RF)	> 94% [27]	Tabular (Features)	Moderate	High (Feature Importance)
Support Vector	~93% [27]	Tabular	Moderate to	Low to

Machine (SVM)		(Features)	High	Moderate
K-Nearest Neighbors (KNN)	~89% [26]	Tabular (Features)	Low (Training), High (Inference)	Low
Convolutional Neural Network (CNN)	> 97% [31]	Time-Series / Image	High	Low (Requires specialized techniques)

Table 2: Comparative Analysis of Classification Models for Exoplanet Detection. This table summarizes the key characteristics of the candidate models based on performance reported in the literature. It clearly shows that for the tabular feature data used in this project, Random Forest offers the best combination of high accuracy and high interpretability, justifying its selection as the primary model.

Defining Success: Evaluation Metrics

Evaluating a classifier on an imbalanced dataset like this requires a nuanced approach that goes beyond simple accuracy. The following metrics will be used to provide a comprehensive assessment of the model's performance.

- **Accuracy:** The proportion of total correct predictions. While easy to calculate, it can be highly misleading in imbalanced scenarios. A model that predicts "FALSE POSITIVE" for every observation could achieve >90% accuracy but would be scientifically useless.[20] It will be reported as a baseline but not used for primary model selection.
- **Precision, Recall, and F1-Score:** These metrics provide a much clearer picture of performance for each class.
  - **Precision** (for the "CONFIRMED" class) measures the proportion of objects predicted as confirmed that are actually confirmed. High precision means a low false positive rate.
  - **Recall** (or Sensitivity/True Positive Rate) measures the proportion of actual confirmed planets that the model correctly identifies. High recall means a low false negative rate and is arguably the most important metric for a discovery tool, as the primary goal is not to miss any potential exoplanets.[29]
  - **F1-Score** is the harmonic mean of precision and recall, providing a single score that balances both concerns.

- **Confusion Matrix:** A table that visualizes the performance of the classifier by showing the counts of true positives, true negatives, false positives, and false negatives for each class. It is essential for understanding the specific types of errors the model is making (e.g., is it confusing CANDIDATE with FALSE POSITIVE?).
- **AUC-ROC Curve (Area Under the Receiver Operating Characteristic Curve):** The ROC curve plots the true positive rate (Recall) against the false positive rate at various classification thresholds. The AUC represents the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative one. It provides an aggregate measure of performance across all possible thresholds and is a robust metric for imbalanced datasets.

By focusing on Recall, F1-Score, and the AUC, we can optimize and select a model that is not just accurate on paper but is genuinely effective at the scientific task of identifying promising exoplanet candidates while minimizing the number of missed discoveries.

Implementation, Training, and Validation of the Exoplanet Classifier

This section provides a practical, step-by-step guide for implementing, training, and rigorously evaluating the recommended Random Forest model. The process leverages standard, robust libraries from the Python data science ecosystem to ensure reproducibility and high performance. The final output of this stage is not just a trained model, but also a deep, quantitative understanding of the factors that drive exoplanet classification.

Setting Up the Python Environment

A standard Python environment is sufficient for this project. The core of the machine learning pipeline will be built using a set of well-established libraries. The following packages are required and can be installed via pip or conda:

- pandas: For data manipulation, including reading CSV files and managing the primary DataFrame.
- numpy: For numerical operations and array manipulation.
- scikit-learn: The cornerstone library for machine learning in Python. It provides implementations of the RandomForestClassifier, preprocessing tools (StandardScaler, IterativeImputer), model selection utilities (GridSearchCV, StratifiedKFold), and evaluation

metrics.
- matplotlib & seaborn: For data visualization, including plotting the confusion matrix and feature importances.
- imblearn: A specialized library for handling imbalanced datasets, providing the SMOTE implementation.

Model Training and Hyperparameter Tuning

Once the data has been fully preprocessed according to the pipeline in Section 2, the dataset is split into a training set and a held-out test set (typically an 80/20 split). It is crucial to use stratification during the split to ensure that the proportion of each class (CONFIRMED, CANDIDATE, FALSE POSITIVE) is the same in both the training and test sets.

The RandomForestClassifier from scikit-learn is then instantiated. While the default parameters often provide a strong baseline, performance can be significantly improved through hyperparameter tuning. The most impactful hyperparameters for a Random Forest are:

- n_estimators: The number of decision trees in the forest. More trees generally improve performance and make predictions more stable, but at the cost of increased computation time.
- max_depth: The maximum depth of each individual tree. If too deep, trees can overfit to the training data. If too shallow, they may not capture the complexity of the data.
- min_samples_leaf: The minimum number of samples required to be at a leaf node. This parameter has a smoothing effect on the model and helps prevent overfitting.
- max_features: The number of features to consider when looking for the best split at each node.

To find the optimal combination of these hyperparameters, a systematic search is performed using GridSearchCV from scikit-learn.[30] This utility exhaustively tries all combinations of a predefined grid of parameter values, using cross-validation to evaluate each combination. The combination that yields the best performance (e.g., the highest mean F1-score across the cross-validation folds) is selected for the final model.

Model Validation and Interpretation

After identifying the best hyperparameters, the final model is trained on the entire training dataset. Its true performance is then assessed on the held-out test set, which the model has never seen before. This provides an unbiased estimate of how the model will perform on new, real-world data.

Performance on Test Set: The model's predictions on the test set are compared against the true labels. This evaluation produces the key success metrics:

- A **Confusion Matrix** is generated to visualize the exact number of correct and incorrect predictions for each class.
- A **Classification Report** is printed, providing the precision, recall, and F1-score for each class, along with their weighted averages. This report is the primary tool for judging the model's effectiveness, with a particular focus on the recall for the CONFIRMED class.
- The **AUC-ROC score** is calculated to provide an aggregate measure of class separability.

Feature Importance Analysis: A key advantage of the Random Forest model is its ability to provide a quantitative measure of feature importance. After training, the feature_importances_ attribute of the classifier object contains a score for each feature, representing its relative contribution to reducing impurity (e.g., Gini impurity) across all trees in the forest.

These importances are extracted and visualized as a horizontal bar chart, ranking the features from most to least predictive. This analysis is not merely a technical step; it is a form of automated scientific discovery. It directly answers the user's query about which variables most significantly impact the final classification decision. The resulting ranking can confirm existing astrophysical intuition (e.g., that transit depth is important) and potentially reveal new, non-obvious relationships in the data. For instance, if a newly engineered SNR feature ranks highly, it provides strong evidence for the success of that feature engineering strategy. The feature importance list is a scientific result in its own right, highlighting the most diagnostically powerful measurements for distinguishing planets from false positives within the vast Kepler, K2, and TESS datasets.

Rank	Feature Name	Gini Importance (Example)	Description
1	fp_flag_ss	0.145	Stellar Eclipse Flag (indicates V-shaped light curve)
2	fp_flag_co	0.121	Centroid Offset Flag (indicates background

			eclipsing binary)
3	fp_flag_nt	0.108	Not Transit-Like Flag (indicates non-physical signal shape)
4	SNR_transit_depth	0.095	Signal-to-Noise of the transit depth measurement
5	planet_radius_earth	0.072	Radius of the candidate object (in Earth radii)
6	orbital_period_days	0.061	Period of the orbit in days
7	impact_parameter	0.054	Transit impact parameter (0 = central, 1 = grazing)
8	transit_duration_hr	0.049	Duration of the transit in hours
9	insolation_flux_earth	0.038	Insolation flux received by the planet relative to Earth
10	stellar_radius_solar	0.031	Radius of the host star (in Solar radii)
11	stellar_eff_temp_k	0.027	Effective temperature of the host star in Kelvin
12	SNR_orbital_period	0.025	Signal-to-Noise of the orbital period

			measurement
13	stellar_logg_cm_s2	0.021	Stellar surface gravity
14	radius_ratio	0.019	Ratio of planetary radius to stellar radius
15	equilibrium_temp_k	0.015	Equilibrium temperature of the planet in Kelvin
16	stellar_density	0.012	Calculated density of the host star
17	transit_epoch_bjd	0.009	Midpoint time of the first transit
18	fp_flag_ec	0.007	Ephemeris Match Flag
19	planetary_density	0.005	Calculated density of the planet (where mass is available)
20	SNR_transit_duration	0.004	Signal-to-Noise of the transit duration measurement

Table 3: Top 20 Predictive Features for Exoplanet Classification (Illustrative Example). This table presents a hypothetical but plausible ranking of feature importances derived from a trained Random Forest model. It demonstrates the immense diagnostic power of the NASA-provided false positive flags, which occupy the top ranks. It also validates the feature engineering strategy, showing that a created feature like SNR_transit_depth can be more predictive than many of the original raw parameters. This table provides a direct, data-driven answer to which variables are most crucial for classification.

Architectural Blueprint for an Interactive Analysis Platform

A trained machine learning model, while powerful, is of limited utility if it remains an inaccessible script. To transform the exoplanet classifier into a practical tool for scientific discovery, it must be embedded within a well-designed, interactive web application. This section outlines the architectural blueprint for such a platform, focusing on a modern technology stack and a user experience (UX) tailored to the needs of researchers and citizen scientists. The goal is to create not just a data display, but a decision-support system that presents a compelling, evidence-based argument for each classification.

System Architecture: A Modern Web Stack

A robust and scalable architecture is essential for deploying the ML model and serving a dynamic user interface. A microservices-based approach using a Python backend and a JavaScript frontend is recommended.

Backend: A lightweight Python web framework is ideal for this application, as it allows for seamless integration with the scikit-learn model. **FastAPI** is the recommended choice.[27] It is a modern, high-performance framework that automatically generates interactive API documentation (via Swagger UI), which is invaluable for development and potential future integrations. The backend will be responsible for:

1. Loading the pre-trained Random Forest model and the feature scaler.
2. Exposing a REST API endpoint (e.g., /predict) that accepts new candidate data in JSON format.
3. Executing the full data preprocessing pipeline on the input data (handling missing values, applying scaling).
4. Making a prediction using the trained model, including generating class probabilities.
5. Returning the classification result, confidence score, and other relevant metadata to the frontend.

Frontend: A modern JavaScript framework is necessary to build a responsive and interactive single-page application (SPA). **React** is a strong choice due to its component-based architecture and vast ecosystem. The frontend will handle all aspects of the user interface, including data input forms, results display, and interactive visualizations.

Data Visualization: The ability to visualize data is central to the platform's utility. A powerful JavaScript charting library that supports interactivity is critical. **Plotly.js** is highly recommended for this purpose.[37] It is built on top of D3.js but offers a higher-level declarative API, making it easier to create a wide range of sophisticated, interactive plots (scatter plots, bar charts, etc.) that support zooming, panning, and hover tooltips—all essential features for scientific data exploration.[38]

User Interface (UI) and User Experience (UX) Design for Scientific Discovery

The design of the user interface must be guided by the principles of clarity, context, and interactivity, drawing from best practices for scientific dashboards.[40] The goal is to minimize cognitive load and allow the user to quickly interpret complex information.[43] A multi-panel dashboard layout is proposed, inspired by professional astronomical analysis tools like AstronomicAL.[44]

Panel 1: Data Input & Control:

- A clean, intuitive form will serve as the primary entry point. It will offer two modes of interaction:
  1. **Manual Entry:** A series of labeled input fields for a user to manually enter the key parameters of a single object of interest (e.g., orbital period, transit depth, stellar radius).
  2. **File Upload:** A drag-and-drop or file selection interface for uploading a CSV file containing one or more candidates that conform to the required input schema.
- A prominent "Analyze" button will trigger the API call to the backend.

Panel 2: Classification Results & Interpretation:

- **Primary Result:** Upon receiving a response from the backend, this panel will clearly and unambiguously display the model's prediction (e.g., "CONFIRMED PLANET," "CANDIDATE," or "FALSE POSITIVE") using a color-coded system (e.g., green for confirmed, yellow for candidate, red for false positive).
- **Confidence Score:** A probability score will be displayed (e.g., "Confidence: 98.7%"), derived from the model's predict_proba method, to indicate the model's certainty in its classification.
- **Prediction Justification:** This is the most critical feature for building user trust. The UI will display the **"Top 5 Most Influential Features"** for that specific prediction. This information can be derived on the backend (e.g., using SHAP values or by analyzing tree

paths for the specific prediction) and presented as a simple ranked list. For example: "This classification was primarily driven by: 1. A very low Stellar Eclipse Flag (0), 2. A high Transit Depth SNR (35.2), 3. A planetary radius in the 'Super-Earth' range (2.1)..." This transforms the model from a black box into an explainable system.

Panel 3: Interactive Visualizations for Context:

- This panel provides the visual evidence to support the classification. It will feature a set of linked, interactive plots generated with Plotly.js:
  - **Contextual Scatter Plot:** A scatter plot of two key features, such as Planet Radius vs. Orbital Period. The plot will show the distribution of all objects in the training set, color-coded by their final disposition. The user's newly analyzed candidate will be plotted as a distinct, highlighted point, allowing them to immediately see where it falls in relation to thousands of known objects. Dropdown menus will allow the user to change the axes to explore other feature relationships (e.g., Transit Duration vs. Transit Depth).
  - **Feature Comparison Bar Charts:** A series of bar charts comparing the user's candidate values for key features (e.g., transit_depth_ppm, impact_parameter) against the average values for the CONFIRMED, CANDIDATE, and FALSE POSITIVE classes from the training data. This provides a quick visual check for how "typical" the candidate's parameters are for each class.
  - **(Optional) Light Curve Plot:** If the platform is extended to accept raw light curve data, this area would display an interactive plot of the phase-folded light curve, allowing the user to visually inspect the transit signal that led to the derived parameters.

Advanced User Interaction: A Human-in-the-Loop System

To ensure the long-term viability and continuous improvement of the platform, an advanced feedback mechanism can be implemented, transforming the system into a collaborative, human-in-the-loop ecosystem. This concept is inspired by the success of citizen science platforms like Zooniverse, which leverage collective human intelligence to classify vast datasets.[46]

- **Feedback Mechanism:** For authenticated users (e.g., registered researchers), the results panel will include simple "Agree" and "Disagree" buttons next to the model's prediction. If a user disagrees, a dialog box can prompt them for the corrected classification and an optional comment.
- **Data Collection for Retraining:** This feedback is not just for UI purposes; it is captured and stored in a separate database. Each entry would log the input features, the model's original prediction, the user's corrected label, and a timestamp.

- **Model Evolution:** Over time, this user-generated feedback creates a new, high-quality labeled dataset. This dataset can be used to periodically retrain and fine-tune the machine learning model. This creates a virtuous cycle: the model helps scientists vet candidates, and the scientists' expert feedback, in turn, helps improve the model. This transforms the platform from a static tool into a dynamic, evolving research ecosystem that becomes more accurate and reliable with use.

Advanced Capabilities and Future Research Directions

The framework detailed in the preceding sections establishes a robust and functional system for exoplanet classification. However, its true potential lies in its extensibility. This section outlines a strategic roadmap for future development, focusing on enhancing the model's analytical power, automating its improvement, and expanding its scope from simple detection to comprehensive characterization. These advancements would position the platform at the forefront of automated exoplanet science.

From Tabular Data to Raw Light Curves: Implementing a CNN

The current model relies on pre-processed tabular data, which is effective but discards the rich morphological information contained within the raw stellar light curves. The next major evolution of the platform should be the integration of a model that can analyze this raw time-series data directly. A **1D Convolutional Neural Network (CNN)** is the state-of-the-art for this task.[31]

The implementation would involve a new data processing pipeline built using Python libraries like lightkurve, which is specifically designed to search, download, and manipulate data from Kepler, K2, and TESS.[11] The key steps would include:

1. **Data Ingestion:** Fetching the raw light curve data (in FITS format) for a given Kepler ID or TIC ID from an archive like MAST.[48]
2. **Preprocessing:** Applying a series of standard astronomical data processing techniques to prepare the light curve for the CNN [49]:
   - **Detrending:** Removing long-term stellar variability and instrumental trends using methods like the Savitzky-Golay filter or by fitting and dividing out a polynomial.
   - **Phase-Folding:** Using the known orbital period of the candidate, the light curve is "folded" so that all transits are overlaid. This dramatically increases the signal-to-noise ratio of the transit event.

    - **Binning and Normalization:** The phase-folded light curve is binned into a fixed-length 1D vector (e.g., 201 data points). The flux is then normalized, typically by setting the out-of-transit median to zero and the maximum transit depth to -1.
3. **Model Training:** This standardized 1D vector, which represents the average shape of the transit, is then used as the input to a 1D CNN. The network learns to identify the characteristic U-shape of a planetary transit and distinguish it from the V-shape of an eclipsing binary or other instrumental artifacts.

Ensemble Modeling for Enhanced Robustness

Once both the Random Forest model (trained on tabular features) and the CNN model (trained on light curve morphology) are operational, their predictive power can be combined through **ensemble modeling**. This approach leverages the strengths of both models, as they learn from different representations of the same underlying phenomenon.

- The Random Forest excels at interpreting the complex, multi-dimensional relationships between the derived physical parameters.
- The CNN excels at recognizing the subtle morphological patterns and shapes within the light curve itself.

A simple yet effective ensemble method would be a **weighted average** of the prediction probabilities from both models. A more sophisticated approach would be **stacking**, where a simple meta-model (e.g., Logistic Regression) is trained to make a final prediction based on the outputs of the RF and CNN models as its inputs. This hybrid approach has the potential to achieve higher accuracy and a lower false positive rate than either model in isolation.

Continuous Integration and Deployment (CI/CD) for Model Retraining

The human-in-the-loop feedback mechanism described in Section 5 generates a continuous stream of high-quality labeled data. To leverage this effectively, a **Continuous Integration and Continuous Deployment (CI/CD)** pipeline should be established. This automated workflow would:

1. **Trigger Periodically:** On a set schedule (e.g., monthly), the pipeline would automatically trigger.
2. **Aggregate Feedback:** It would collect all new, validated user feedback from the database.
3. **Retrain Model:** The new data would be combined with the existing training set, and the

entire model training and hyperparameter tuning process would be re-run.

4. **Validate Performance:** The newly trained model would be automatically evaluated against a benchmark dataset. If its performance (e.g., F1-score) shows a statistically significant improvement over the currently deployed model, it is flagged for promotion.
5. **Deploy New Model:** With a single approval step, the new, improved model can be deployed to the production backend, replacing the old one.

This CI/CD pipeline ensures that the system is not static but continuously learns and improves over time, becoming more accurate as more experts use it.

Expanding the Scope: Characterization Beyond Detection

The ultimate goal of exoplanet science is not just detection but characterization. The platform can be extended to support this goal by incorporating unsupervised machine learning techniques to explore the population of confirmed planets.

Using the rich feature set of the confirmed exoplanets, a clustering algorithm like **K-Means** or a dimensionality reduction technique like **Uniform Manifold Approximation and Projection (UMAP)** can be applied.[50] These algorithms can automatically identify distinct groups or clusters of planets within the high-dimensional feature space. These data-driven clusters could correspond to known astrophysical classes (e.g., Hot Jupiters, Super-Earths, Mini-Neptunes) or potentially reveal new, previously unrecognized sub-populations of exoplanets.

This functionality could be integrated into the web interface as a "Population Explorer" tool. Users could visualize the confirmed exoplanet population in a 2D or 3D interactive plot generated by UMAP, color-coded by the discovered clusters. This would transform the platform from a classification tool for individual objects into a powerful instrument for population-level discovery, enabling scientists to explore the vast diversity of planetary systems in our galaxy.

Conclusion

This report has detailed a comprehensive, multi-stage framework for the development of an advanced exoplanet detection and analysis platform. By integrating established scientific principles with modern machine learning techniques and user-centric web design, this system moves beyond simple classification to provide a robust, interpretable, and extensible tool for astronomical research.

The project begins with a rigorous approach to data management, acknowledging the heterogeneity of NASA's Kepler, K2, and TESS datasets. The proposed data harmonization schema and the sophisticated preprocessing pipeline—including model-based imputation, outlier detection, and extensive feature engineering—are designed to create a high-quality, feature-rich dataset that embeds physical intuition directly into the data. This foundational work is critical for the success of any subsequent modeling.

The selection of a Random Forest classifier as the primary model is justified by its consistently high performance in the literature, its robustness, and its crucial interpretability. By leveraging the model's ability to rank feature importance, the system can provide not just a prediction but an explanation, a feature that is paramount for building trust and utility within a scientific context. The emphasis on metrics like recall and F1-score ensures the model is optimized for the primary scientific goal: minimizing missed discoveries.

The architectural blueprint for the web platform transforms the model into a dynamic, interactive tool. The proposed multi-panel dashboard is designed to present a clear, evidence-based argument for each classification, empowering users to understand the model's reasoning through contextual visualizations. The inclusion of a human-in-the-loop feedback mechanism provides a strategic vision for the platform's long-term evolution, allowing it to become a collaborative ecosystem that continuously improves as it is used by the scientific community.

Finally, the roadmap for future development—incorporating CNNs for raw light curve analysis, building ensemble models, automating retraining through CI/CD pipelines, and expanding into unsupervised characterization—ensures the project's lasting relevance. By following this framework, it is possible to construct a powerful platform that not only automates the laborious task of vetting exoplanet candidates but also serves as a catalyst for new discoveries, helping to uncover the secrets of the thousands of worlds hidden in NASA's vast astronomical archives.

Works cited

1. Optical Detectors for the Discovery of Transiting Exoplanets - Andor Technology, accessed on September 30, 2025, https://andor.oxinst.com/learning/view/article/optical-detectors-for-the-discovery-of-transiting-exoplanets
2. The Detection and Characterization of Extrasolar Planets - MDPI, accessed on September 30, 2025, https://www.mdpi.com/2078-1547/5/2/296
3. What's a transit? - NASA Science, accessed on September 30, 2025, https://science.nasa.gov/exoplanets/whats-a-transit/
4. Down in Front!: The Transit Photometry Method | The Planetary Society, accessed on September 30, 2025,

https://www.planetary.org/articles/down-in-front-the-transit-photometry-method

5. Exoplanet Photometry and False Positives – Journal of Research in Progress Vol. 3, accessed on September 30, 2025, https://pressbooks.howardcc.edu/jrip3/chapter/exoplanet-photometry-and-false-positives/
6. Transit Light Curve Tutorial - Andrew Vanderburg, accessed on September 30, 2025, https://avanderburg.github.io/tutorial/tutorial.html
7. Transit Method - Las Cumbres Observatory, accessed on September 30, 2025, https://lco.global/spacebook/exoplanets/transit-method/
8. Kepler Objects of Interest (KOI).csv
9. Exoplanet validation with machine learning: 50 new validated Kepler planets | Monthly Notices of the Royal Astronomical Society | Oxford Academic, accessed on September 30, 2025, https://academic.oup.com/mnras/article/504/4/5327/5894933
10. A World Away: Hunting for Exoplanets with AI - NASA Space Apps Challenge, accessed on September 30, 2025, https://www.spaceappschallenge.org/2025/challenges/a-world-away-hunting-for-exoplanets-with-ai/
11. TESS Data Analysis Tools - HEASARC - NASA, accessed on September 30, 2025, https://heasarc.gsfc.nasa.gov/docs/tess/data-analysis-tools.html
12. Revisiting the Kepler field with TESS: Improved ephemerides using TESS 2 min data | Monthly Notices of the Royal Astronomical Society | Oxford Academic, accessed on September 30, 2025, https://academic.oup.com/mnras/article/503/3/4092/6166781
13. Imputation of missing photometric data and photometric redshift estimation for CSST | Monthly Notices of the Royal Astronomical Society | Oxford Academic, accessed on September 30, 2025, https://academic.oup.com/mnras/article/531/3/3539/7688471
14. A Guide to Handling Missing values in Python - Kaggle, accessed on September 30, 2025, https://www.kaggle.com/code/parulpandey/a-guide-to-handling-missing-values-in-python
15. How to Handle Time Series Missing Data | Towards Data Science, accessed on September 30, 2025, https://towardsdatascience.com/how-to-handle-time-series-missing-data-d45e9aaae72c/
16. How to Handle Missing Data with Python - MachineLearningMastery.com, accessed on September 30, 2025, https://machinelearningmastery.com/handle-missing-data-python/
17. Detecting outliers in astronomical images with deep generative networks - Oxford Academic, accessed on September 30, 2025, https://academic.oup.com/mnras/article/496/2/2346/5858005
18. Outlier Detection Techniques for Time Series | by Alex Eslava - Medium, accessed on September 30, 2025,

https://medium.com/@alex.eslava96/outlier-detection-techniques-for-time-series-9868db2875c2
19. An algorithm for the fitting of planet models to Kepler light curves, accessed on September 30, 2025, https://ntrs.nasa.gov/api/citations/20180007239/downloads/20180007239.pdf
20. Applying Machine Learning to Exoplanet Detection - Charles Zhao, accessed on September 30, 2025, https://hczhao.me/static/portfolio/docs/exoplanet.pdf
21. Detecting Exoplanets from Light Curves of Kepler Mission | Towards Data Science, accessed on September 30, 2025, https://towardsdatascience.com/detecting-exoplanets-from-light-curves-of-kepler-mission-a1f2a3f667fb/
22. PHYS037 - Exoplanet Classification With Machine Learning | Regeneron ISEF 2025, accessed on September 30, 2025, https://isef.net/project/phys037-exoplanet-classification-with-machine-learning
23. Identification and Classification of Exoplanets using Light Intensity. - NORMA@NCI Library, accessed on September 30, 2025, https://norma.ncirl.ie/6122/1/aadityabalkrishnagarude.pdf
24. Advanced Modeling for Exoplanet Detection and Characterization - arXiv, accessed on September 30, 2025, https://arxiv.org/pdf/2506.17665
25. Machine Learning Pipeline for Exoplanet Classification - SMU Scholar, accessed on September 30, 2025, https://scholar.smu.edu/cgi/viewcontent.cgi?article=1070&context=datasciencereview
26. [2508.09689] Machine Learning for Exoplanet Detection: A Comparative Analysis Using Kepler Data - arXiv, accessed on September 30, 2025, https://arxiv.org/abs/2508.09689
27. Evaluating Classification Algorithms: Exoplanet Detection using Kepler Time Series Data, accessed on September 30, 2025, https://arxiv.org/html/2402.15874v1
28. Exoplanet Detection using Machine Learning - IJSREM Journal, accessed on September 30, 2025, https://ijsrem.com/download/exoplanet-detection-using-machine-learning/
29. Detecting exoplanets with machine learning - DiVA portal, accessed on September 30, 2025, http://www.diva-portal.org/smash/get/diva2:1325376/FULLTEXT01.pdf
30. bhavinidata/ExoplanetExplorationML: Classify candidate exoplanets using various machine learning models like Random Forest, KNN, Logistic Regression and SVM - GitHub, accessed on September 30, 2025, https://github.com/bhavinidata/ExoplanetExplorationML
31. Deep learning exoplanets detection by combining real and synthetic ..., accessed on September 30, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC9132280/
32. One-Dimensional Convolutional Neural Networks for Detecting Transiting Exoplanets, accessed on September 30, 2025, https://www.mdpi.com/2075-1680/12/4/348
33. One-dimensional Convolutional Neural Networks for Detecting Transiting Exoplanets - arXiv, accessed on September 30, 2025,

https://arxiv.org/abs/2312.07161
34. [2402.15874] Evaluating Classification Algorithms: Exoplanet Detection using Kepler Time Series Data - arXiv, accessed on September 30, 2025, https://arxiv.org/abs/2402.15874
35. Assessment of Ensemble-Based Machine Learning Algorithms for Exoplanet Identification, accessed on September 30, 2025, https://www.mdpi.com/2079-9292/13/19/3950
36. Exoplanet Detection Using Machine Learning Models Trained on Synthetic Light Curves, accessed on September 30, 2025, https://arxiv.org/html/2507.19520v1
37. Plotly JavaScript Open Source Graphing Library, accessed on September 30, 2025, https://plotly.com/javascript/
38. 20 Must-Know JavaScript Libraries for Data Visualization - DEV Community, accessed on September 30, 2025, https://dev.to/web_dev-usman/20-must-know-javascript-libraries-for-data-visualization-508d
39. D3 by Observable | The JavaScript library for bespoke data visualization, accessed on September 30, 2025, https://d3js.org/
40. Designing an intuitive user interface (UI) for data researchers working with complex datasets is essential to enhance their ability to extract meaningful insights rapidly. An effective UI reduces cognitive load, offers clear visual cues, and supports seamless interaction with data, making complex information accessible and actionable. Below are the best practices to achieve this goal. - Zigpoll, accessed on September 30, 2025, https://www.zigpoll.com/content/what-are-the-best-practices-for-designing-an-intuitive-user-interface-that-helps-data-researchers-quickly-interpret-complex-datasets
41. Data visualization UI: best practices and winning approaches - Transcenda, accessed on September 30, 2025, https://www.transcenda.com/insights/data-visualization-ui-best-practices-and-winning-approaches
42. Effective Dashboard Design Principles for 2025 - UXPin, accessed on September 30, 2025, https://www.uxpin.com/studio/blog/dashboard-design-principles/
43. The Ultimate Data Visualization Handbook for Designers - UX Magazine, accessed on September 30, 2025, https://uxmag.com/articles/the-ultimate-data-visualization-handbook-for-designers
44. Stevens, G., Fotopoulou, S., Bremer, M., & Ray, O. (2021). AstronomicAL: an interactive dashboard for visualisation, integra - University of Bristol Research Portal, accessed on September 30, 2025, https://research-information.bris.ac.uk/files/288081752/Full_text_PDF_final_published_version_.pdf
45. AstronomicAL: an interactive dashboard for visualisation, integration and classification of data with Active Learning - Journal of Open Source Software, accessed on September 30, 2025, https://joss.theoj.org/papers/10.21105/joss.03635.pdf

46. Zooniverse, accessed on September 30, 2025, https://www.zooniverse.org/
47. Tutorials — Lightkurve - GitHub Pages, accessed on September 30, 2025, https://lightkurve.github.io/lightkurve/tutorials/index.html
48. Using Kepler Data to Plot a Light Curve, accessed on September 30, 2025, https://spacetelescope.github.io/notebooks/notebooks/MAST/Kepler/Kepler_Lightcurve/kepler_lightcurve.html
49. Detection of Exoplanets in Transit Light Curves with Conditional Flow Matching and XGBoost - MDPI, accessed on September 30, 2025, https://www.mdpi.com/2079-9292/14/9/1738
50. Classifying Exoplanets with Machine Learning - Copernicus Meetings, accessed on September 30, 2025, https://presentations.copernicus.org/EPSC2020/EPSC2020_833_presentation.pdf