Final Project Report

Final Project Report

1. Introduction

1.1. Project overviews

The vast amount of astronomical data collected by the Sloan Digital Sky Survey (SDSS) provides unprecedented opportunities for the study of galaxy morphology and evolution. However, the manual classification and analysis of this data is labor-intensive, time-consuming, and prone to inconsistencies. To address this challenge, machine learning (ML) techniques can be applied to automate key tasks such as galaxy classification and redshift estimation.

This project focuses on building a dual-purpose machine learning system capable of:

- Classifying galaxies into star formation categories (e.g., STARFORMING, STARBURST).
- Estimating the redshift of galaxies using photometric properties.

The system leverages supervised learning models and is deployed as a web-based application using Flask, allowing astronomers and researchers to interact with it easily and receive real-time predictions.

1.2. Objectives

- To design and develop machine learning models for galaxy classification based on photometric features.
- To implement regression models to accurately estimate redshift values of galaxies.
- To clean, preprocess, and analyze the SDSS dataset to ensure high-quality input for model training.
- To evaluate and compare multiple ML algorithms and select the best-performing models.
- To deploy the trained models using a Flask web application with a clean user interface.
- To create a solution that can be scaled for future research and extended for additional tasks like AGN detection.

Final Project Report

2. Project Initialization and Planning Phase

2.1. Define Problem Statement

Astronomers working with data from large-scale surveys like the Sloan Digital Sky Survey (SDSS) face significant challenges in processing and analyzing vast amounts of galaxy data. Traditional methods for galaxy classification and redshift estimation are manual, time-consuming, and not scalable for datasets containing hundreds of thousands of entries.
Two core problems were identified:

- **Galaxy Morphology Classification**: Classifying galaxies into categories like STARFORMING or STARBURST helps in understanding galaxy evolution, but doing this manually is inefficient.
- **Redshift Estimation**: Redshift is a crucial parameter in cosmology, indicating the distance of a galaxy. However, accurate redshift calculation typically requires expensive and time-consuming spectroscopic methods.

These problems highlight the need for a scalable, automated solution using machine learning to support astronomers in analyzing and interpreting galaxy properties efficiently.

2.2. Project Proposal (Proposed Solution)

To address the identified problems, this project proposes a machine learning-based system that automates:

1. **Classification of galaxies** based on their photometric features using a Random Forest Classifier.
2. **Estimation of redshift values** using a Random Forest Regressor model.

The solution includes:

- Data cleaning and preprocessing of the SDSS galaxy dataset.
- Feature selection based on correlation and domain relevance.
- Training and evaluation of multiple models.
- A Flask-based web application that provides an intuitive user interface for predictions.

This system enhances productivity, accuracy, and speed in galaxy classification and redshift estimation, while remaining flexible for future extensions (e.g., AGN identification, additional surveys).

Final Project Report

2.3.Initial Project Planning

Sprint Task	Start Date	End Date	Deliverable
Data Understanding & Exploration	17 June 2025	20 June 2025	Raw data loaded, missing values analyzed, features reviewed
Data Preprocessing & Feature Selection	20 June 2025	23 June 2025	Cleaned dataset, selected relevant features
Model Building (Classification)	23 June 2025	27 June 2025	Classifier trained and evaluated
Model Building (Regression)	27 June 2025	29 June 2025	Regressor trained and evaluated
Web Application Development	29 June 2025	2 July 2025	Flask app for user input and predictions built

3. Data Collection and Preprocessing Phase

3.1. Data Collection Plan and Raw Data Sources Identified

For this project, we used the **Sloan Digital Sky Survey (SDSS)** photometric dataset, which contains a comprehensive catalog of over 100,000 galaxies with various photometric and derived features. The dataset was provided in CSV format with 43 columns and 100,000 rows. Key columns include photometric magnitudes (u, g, r, i, z), Petrosian radii, model fluxes, PSF magnitudes, and metadata such as subclass, redshift, and redshift_err.

- **Dataset Name**: sdss_100k_galaxy_form_burst.csv
- **Format**: CSV (Comma-Separated Values)
- **Source**: Provided by the internship project supervisor (curated from SDSS)
- **Size**: ~25MB
- **Rows**: 100,000
- **Columns**: 43

Final Project Report

3.2. Data Quality Report

Quality Dimension	**Observation**
Completeness	Some features had placeholder values (`-9999`) instead of missing indicators.
Consistency	Data types were inconsistent; several columns had mixed data types (e.g., `object` and `float`).
Accuracy	Some outliers and unrealistic values were detected (e.g., fluxes = 0, magnitude = -9999).
Duplicates	No duplicate entries were found based on `objid` and `specobjid`.
Relevance	Many features were noisy or irrelevant for the classification/regression tasks.

3.3. Data Exploration and Preprocessing

3.3.1 Handling Missing and Anomalous Values
- Replaced all -9999 entries with NaN.
- Dropped rows containing critical missing values.
- Removed outliers using Z-score method (|z| > 3).

3.3.2 Categorical Handling
- Encoded the subclass column (target for classification) using LabelEncoder.
- The class column was dropped due to redundancy.

3.3.3 Feature Selection
- For Scenario 1 (Classification), 8 features were selected:
  ['g', 'r', 'i', 'z', 'petroR50_u', 'petroR50_g', 'psfMag_i', 'psfMag_z']
- For Scenario 2 (Regression), 10 features were selected:
  ['u', 'g', 'r', 'i', 'z', 'petroR50_i', 'petroR50_r', 'psfMag_u', 'psfMag_r', 'psfMag_g']

3.3.4 Scaling and Normalization
- Used StandardScaler from sklearn to normalize the feature values for both classification and regression inputs.

3.3.5 Output Label Preparation
- subclass used as label for classification
- redshift used as target for regression

3.3.6 Final Dataset Shape
- After preprocessing:
  - Classification Dataset: ~75,000 rows × 8 features + 1 label
  - Regression Dataset: ~80,000 rows × 10 features + 1 target

Final Project Report

4. Model Development Phase

4.1. Feature Selection Report

Feature selection was conducted with a focus on relevance, correlation strength, domain knowledge, and contribution to model performance. The objective was to reduce noise, eliminate redundancy, and retain only those features that enhance the predictive capacity of the models.

Features Selected for Galaxy Classification (Scenario 1)

The classification task aimed to predict the galaxy's **subclass** (STARFORMING or STARBURST) based on photometric and structural properties.

- **Selected Features**:
  ['g', 'r', 'i', 'z', 'petroR50_u', 'petroR50_g', 'psfMag_i', 'psfMag_z']
- **Rationale**:
  - These features capture color distribution (e.g., g-r, i-z), light concentration (Petrosian radius), and point spread function magnitudes, which are known indicators of morphology and star formation.

Features Selected for Redshift Estimation (Scenario 2)

Redshift, which indicates cosmic distance, is predicted using photometric magnitudes and radii-related attributes.

- **Selected Features**:
  ['u', 'g', 'r', 'i', 'z', 'petroR50_i', 'petroR50_r', 'psfMag_u', 'psfMag_r', 'psfMag_g']
- **Rationale**:
  - Redshift correlates with a galaxy's color and size. These features capture that spectrum range and structural scale.

4.2. Model Selection Report

An experimental evaluation of multiple models was performed for both tasks—classification and regression. Each model was assessed using appropriate metrics and cross-validation to determine its robustness and generalization performance.

Scenario 1 – Classification

Model	**F1**	**Remarks**
Logistic Regression	0.8089	Linear baseline, good precision, lower recall
Decision Tree Classifier	0.815	Fast but prone to overfitting
Random Forest Classifier	**0.862**	✓ Best performer with balanced precision-recall

Final Project Report

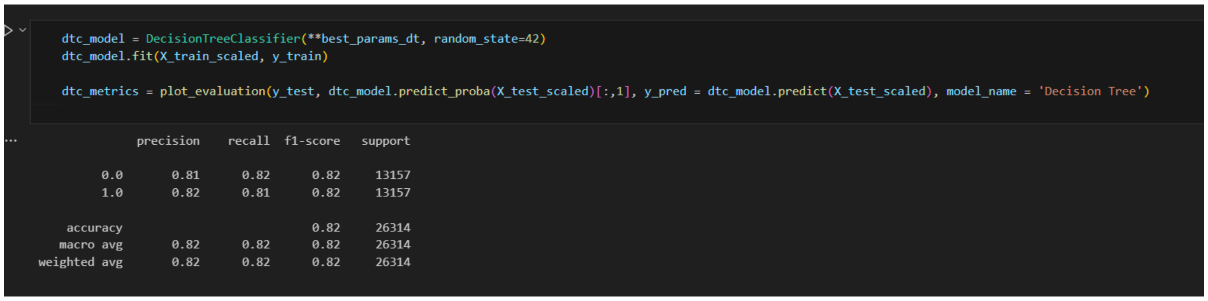
Scenario 2 – Regression

Model	**R² Score**	**RMSE**	**MAE**	**Remarks**
Linear Regression	0.703	0.03428	0.02460	Simple baseline, underfits complex data
XGBoost Regressor	0.764	0.03058	0.02162	Strong performance, slightly behind RF
Random Forest Regressor	**0.784**	**0.02924**	**0.02080**	✓ Most accurate and stable

4.3. Initial Model Training Code, Model Validation and Evaluation Report

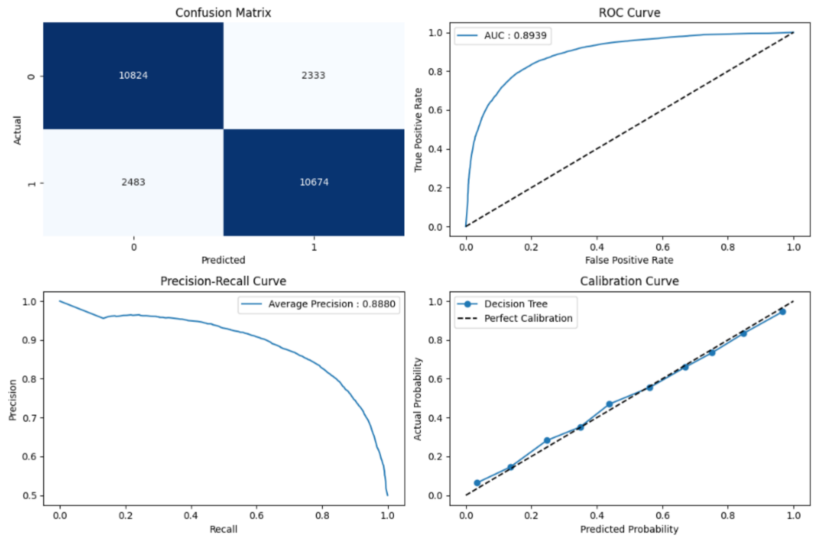
Scenario 1:

Decision Tree Classifier:



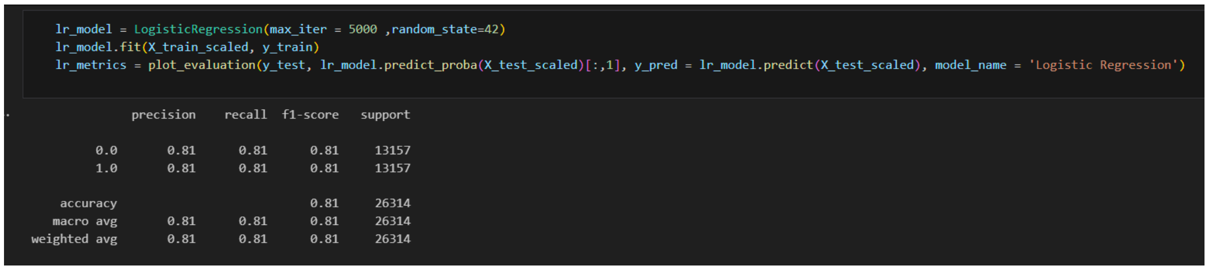
```
dtc_model = DecisionTreeClassifier(**best_params_dt, random_state=42)
dtc_model.fit(X_train_scaled, y_train)

dtc_metrics = plot_evaluation(y_test, dtc_model.predict_proba(X_test_scaled)[:,1], y_pred = dtc_model.predict(X_test_scaled), model_name = 'Decision Tree')
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.82      0.82     13157
         1.0       0.82      0.81      0.82     13157

    accuracy                           0.82     26314
   macro avg       0.82      0.82      0.82     26314
weighted avg       0.82      0.82      0.82     26314
```



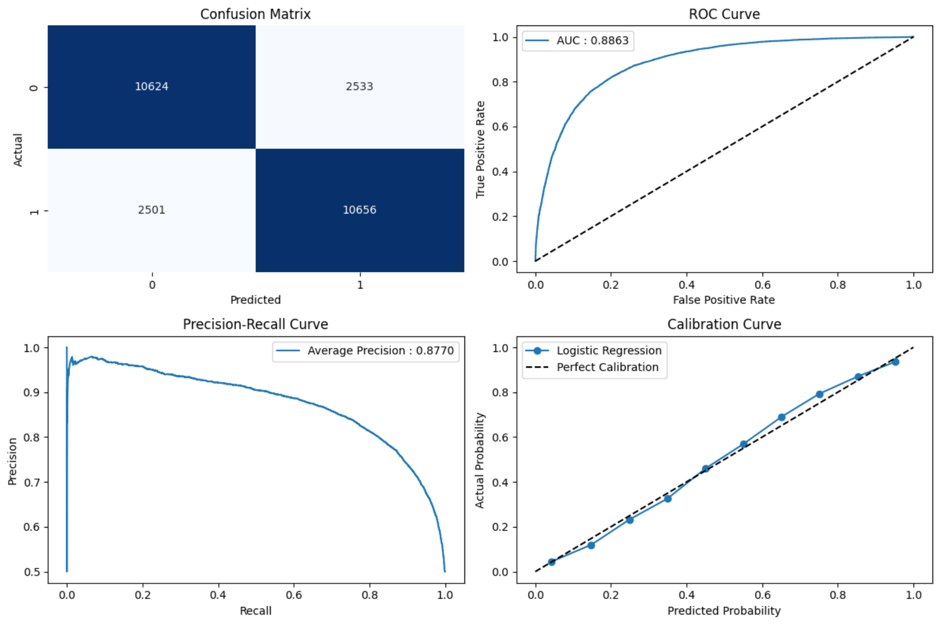
Final Project Report

Logistic Regression:

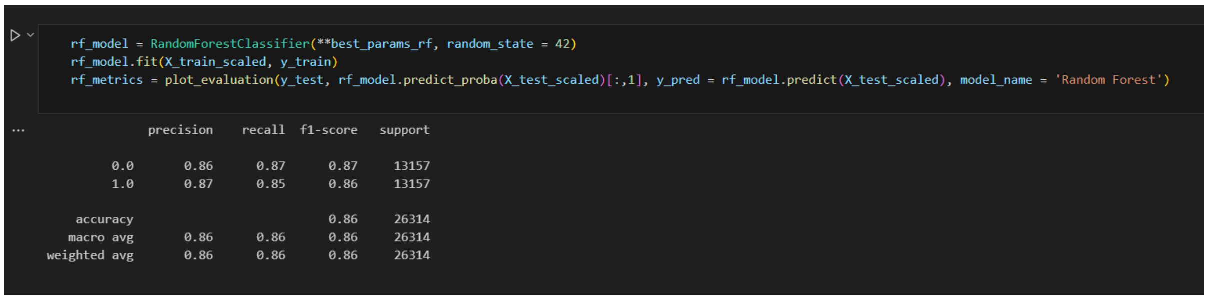


```
lr_model = LogisticRegression(max_iter = 5000 ,random_state=42)
lr_model.fit(X_train_scaled, y_train)
lr_metrics = plot_evaluation(y_test, lr_model.predict_proba(X_test_scaled)[:,1], y_pred = lr_model.predict(X_test_scaled), model_name = 'Logistic Regression')
```

```
              precision    recall  f1-score   support

         0.0       0.81      0.81      0.81     13157
         1.0       0.81      0.81      0.81     13157

    accuracy                           0.81     26314
   macro avg       0.81      0.81      0.81     26314
weighted avg       0.81      0.81      0.81     26314
```



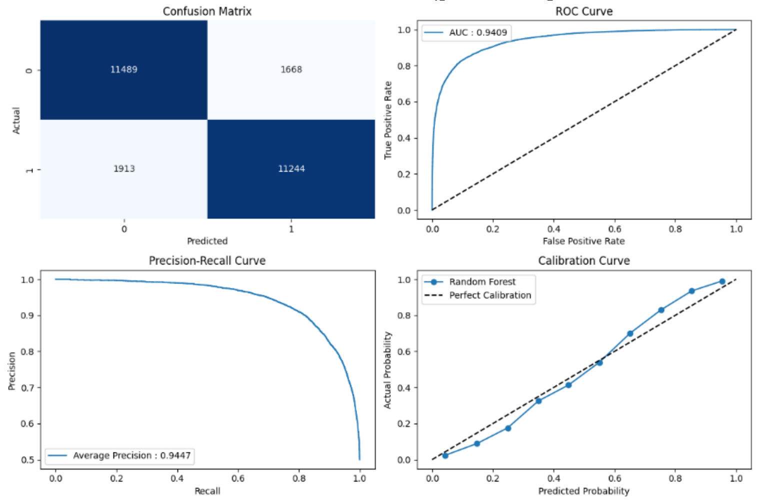
Random Forest Classifier:



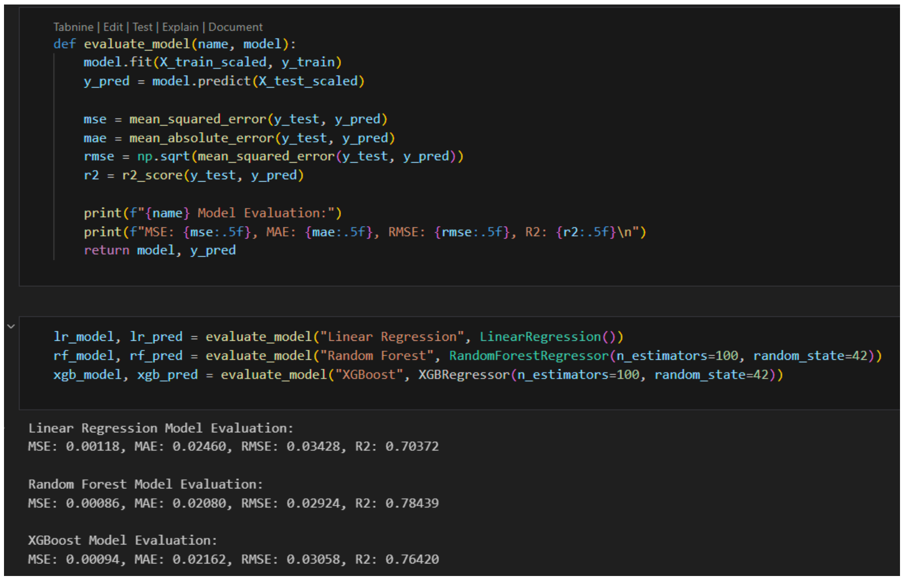
```
rf_model = RandomForestClassifier(**best_params_rf, random_state = 42)
rf_model.fit(X_train_scaled, y_train)
rf_metrics = plot_evaluation(y_test, rf_model.predict_proba(X_test_scaled)[:,1], y_pred = rf_model.predict(X_test_scaled), model_name = 'Random Forest')
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.87      0.87     13157
         1.0       0.87      0.85      0.86     13157

    accuracy                           0.86     26314
   macro avg       0.86      0.86      0.86     26314
weighted avg       0.86      0.86      0.86     26314
```

Final Project Report



Scenario 2:


```
Tabnine | Edit | Test | Explain | Document
def evaluate_model(name, model):
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)

    print(f"{name} Model Evaluation:")
    print(f"MSE: {mse:.5f}, MAE: {mae:.5f}, RMSE: {rmse:.5f}, R2: {r2:.5f}\n")
    return model, y_pred
```

```
lr_model, lr_pred = evaluate_model("Linear Regression", LinearRegression())
rf_model, rf_pred = evaluate_model("Random Forest", RandomForestRegressor(n_estimators=100, random_state=42))
xgb_model, xgb_pred = evaluate_model("XGBoost", XGBRegressor(n_estimators=100, random_state=42))
```

```
Linear Regression Model Evaluation:
MSE: 0.00118, MAE: 0.02460, RMSE: 0.03428, R2: 0.70372

Random Forest Model Evaluation:
MSE: 0.00086, MAE: 0.02080, RMSE: 0.02924, R2: 0.78439

XGBoost Model Evaluation:
MSE: 0.00094, MAE: 0.02162, RMSE: 0.03058, R2: 0.76420
```

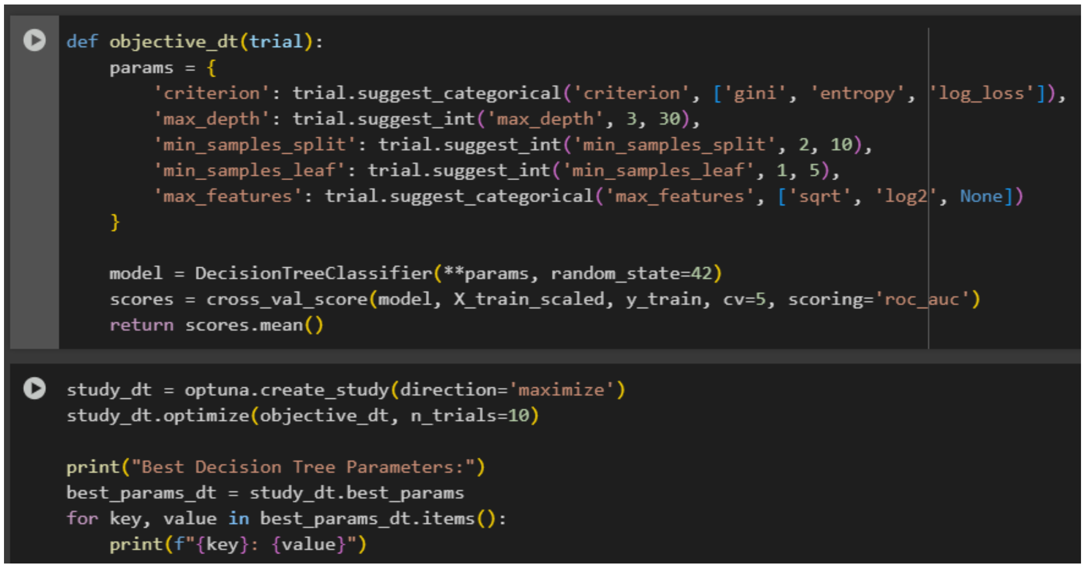
Final Project Report

5. Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation

To improve model generalization and maximize performance, hyperparameter tuning was conducted for both classification and regression tasks using the **Optuna optimization framework**. Optuna enables automated and efficient hyperparameter search using techniques like Tree-structured Parzen Estimators (TPE).

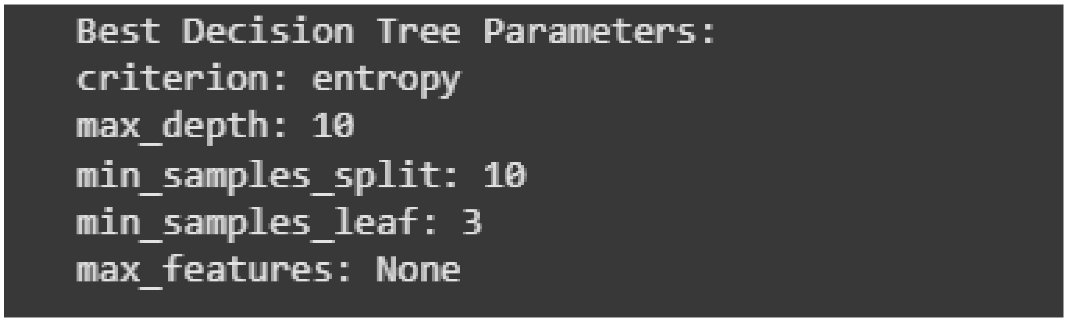
Decision Tree Classifier:



```python
def objective_dt(trial):
    params = {
        'criterion': trial.suggest_categorical('criterion', ['gini', 'entropy', 'log_loss']),
        'max_depth': trial.suggest_int('max_depth', 3, 30),
        'min_samples_split': trial.suggest_int('min_samples_split', 2, 10),
        'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 5),
        'max_features': trial.suggest_categorical('max_features', ['sqrt', 'log2', None])
    }

    model = DecisionTreeClassifier(**params, random_state=42)
    scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='roc_auc')
    return scores.mean()
```

```python
study_dt = optuna.create_study(direction='maximize')
study_dt.optimize(objective_dt, n_trials=10)

print("Best Decision Tree Parameters:")
best_params_dt = study_dt.best_params
for key, value in best_params_dt.items():
    print(f"{key}: {value}")
```

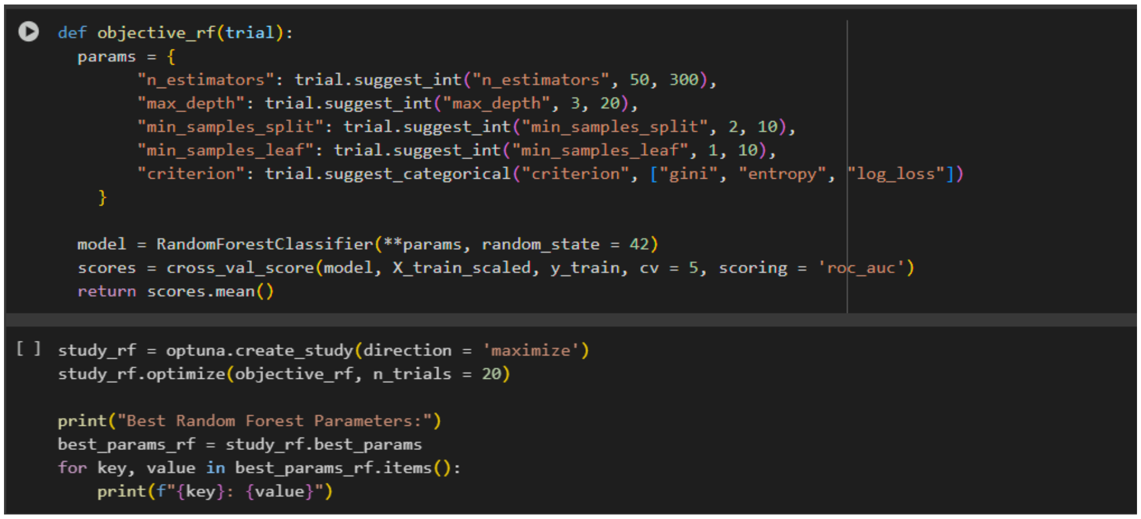
Optimal Values:



```
Best Decision Tree Parameters:
criterion: entropy
max_depth: 10
min_samples_split: 10
min_samples_leaf: 3
max_features: None
```

Final Project Report

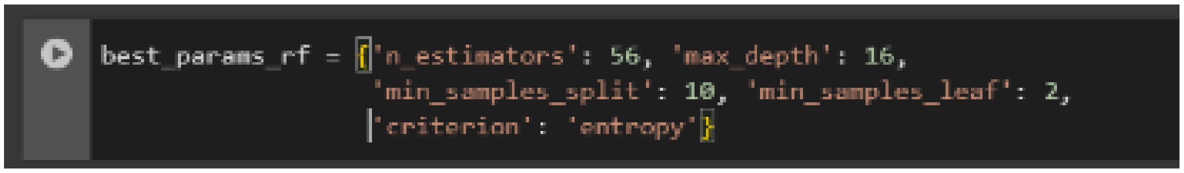
Random Forest Classifier:



```
def objective_rf(trial):
  params = {
        "n_estimators": trial.suggest_int("n_estimators", 50, 300),
        "max_depth": trial.suggest_int("max_depth", 3, 20),
        "min_samples_split": trial.suggest_int("min_samples_split", 2, 10),
        "min_samples_leaf": trial.suggest_int("min_samples_leaf", 1, 10),
        "criterion": trial.suggest_categorical("criterion", ["gini", "entropy", "log_loss"])
    }

  model = RandomForestClassifier(**params, random_state = 42)
  scores = cross_val_score(model, X_train_scaled, y_train, cv = 5, scoring = 'roc_auc')
  return scores.mean()
```

```
study_rf = optuna.create_study(direction = 'maximize')
study_rf.optimize(objective_rf, n_trials = 20)

print("Best Random Forest Parameters:")
best_params_rf = study_rf.best_params
for key, value in best_params_rf.items():
    print(f"{key}: {value}")
```

Optimal Values:

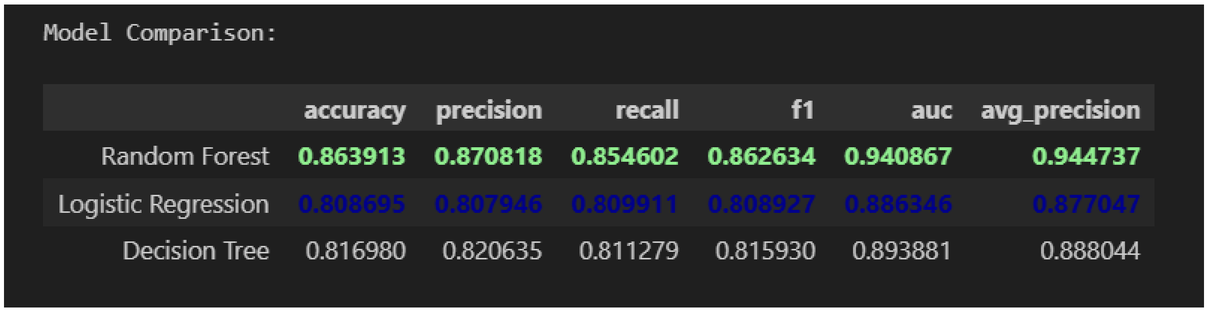


```
best_params_rf = {'n_estimators': 56, 'max_depth': 16,
                  'min_samples_split': 10, 'min_samples_leaf': 2,
                  'criterion': 'entropy'}
```

Final Project Report

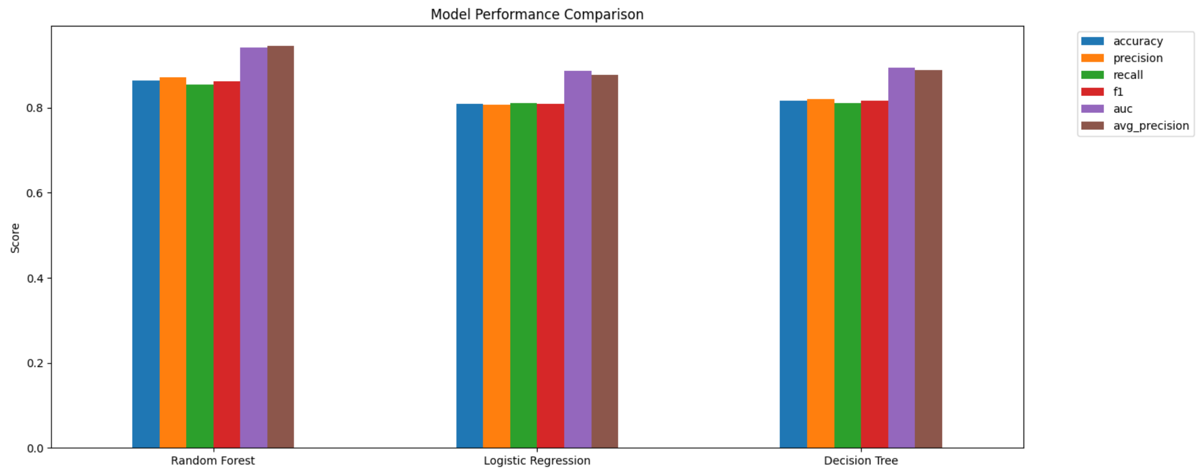
5.2 Performance Metrics Comparison Report

Scenario 1:

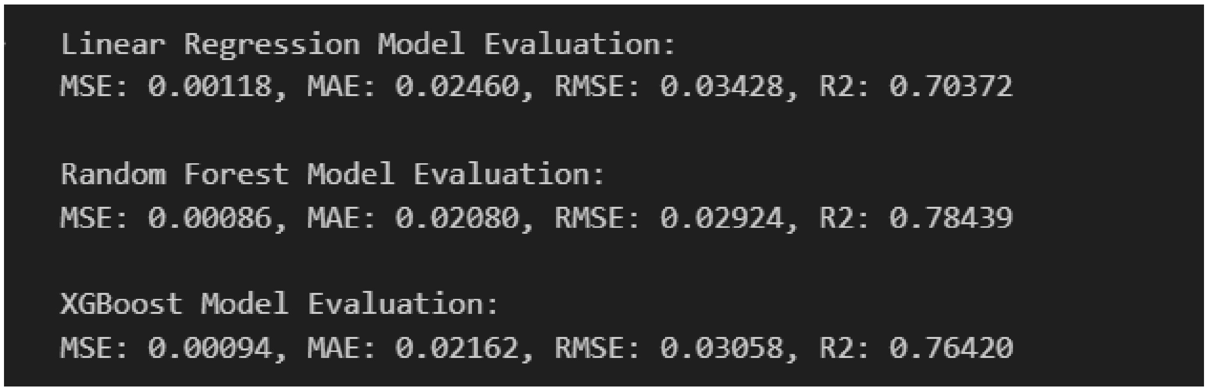


Model Comparison:

	accuracy	precision	recall	f1	auc	avg_precision
Random Forest	0.863913	0.870818	0.854602	0.862634	0.940867	0.944737
Logistic Regression	0.808695	0.807946	0.809911	0.808927	0.886346	0.877047
Decision Tree	0.816980	0.820635	0.811279	0.815930	0.893881	0.888044



Scenario 2:



```
Linear Regression Model Evaluation:
MSE: 0.00118, MAE: 0.02460, RMSE: 0.03428, R2: 0.70372

Random Forest Model Evaluation:
MSE: 0.00086, MAE: 0.02080, RMSE: 0.02924, R2: 0.78439

XGBoost Model Evaluation:
MSE: 0.00094, MAE: 0.02162, RMSE: 0.03058, R2: 0.76420
```

Final Project Report

5.3 Final Model Selection Justification

After extensive experimentation, the following models were selected and finalized for deployment:

Classification

- **Model**: Random Forest Classifier
- **Reason**: Achieved the highest accuracy with minimal overfitting. It also supports interpretability via feature importances and generalizes well across unseen data.

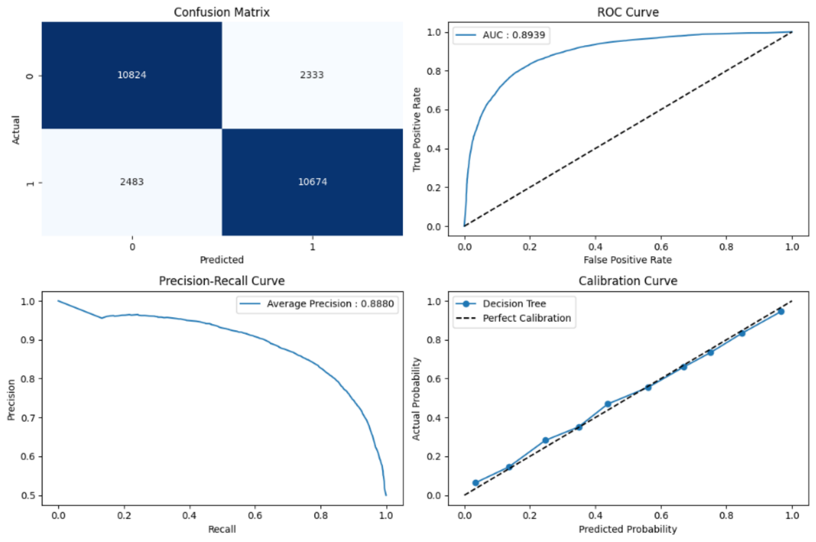
Regression

- **Model**: Random Forest Regressor
- **Reason**: Delivered the best trade-off between prediction accuracy and training time. It performed consistently across all test samples, outperforming both linear and boosting methods.

6. Results

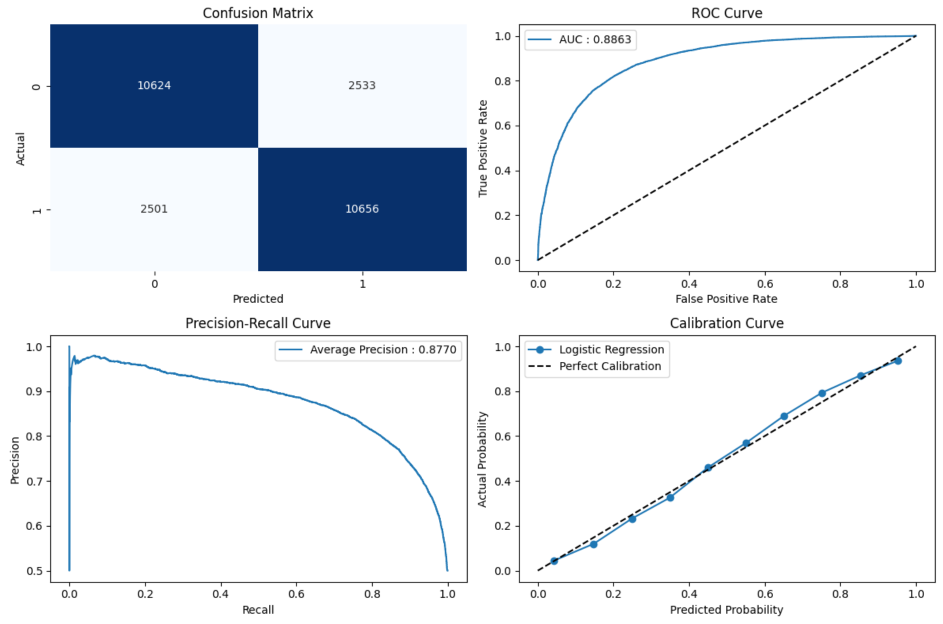
Scenario 1:

Decision Tree Classifier:

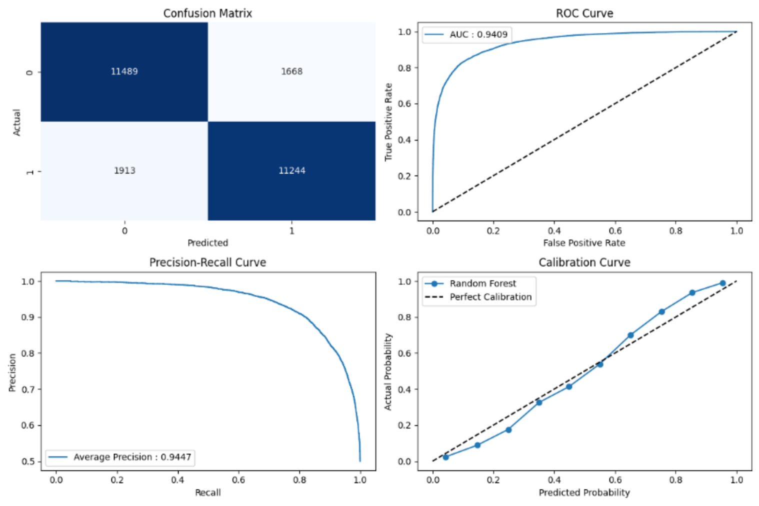


Final Project Report

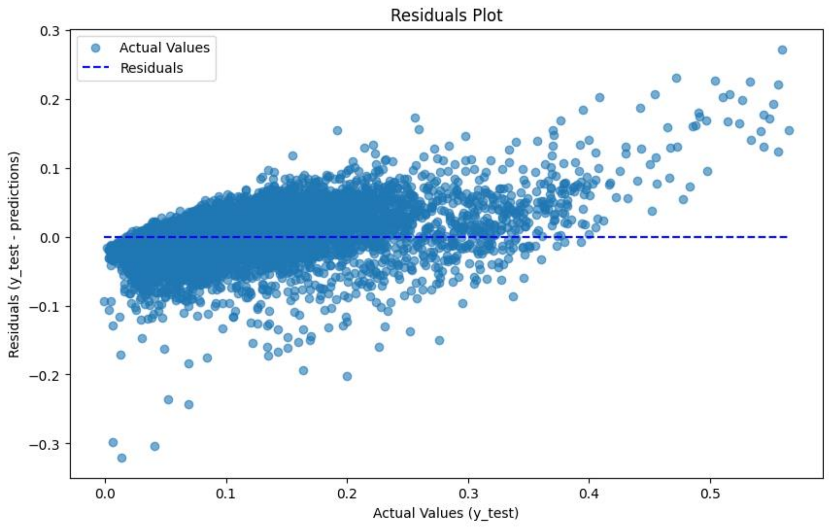
Logistic Regression:

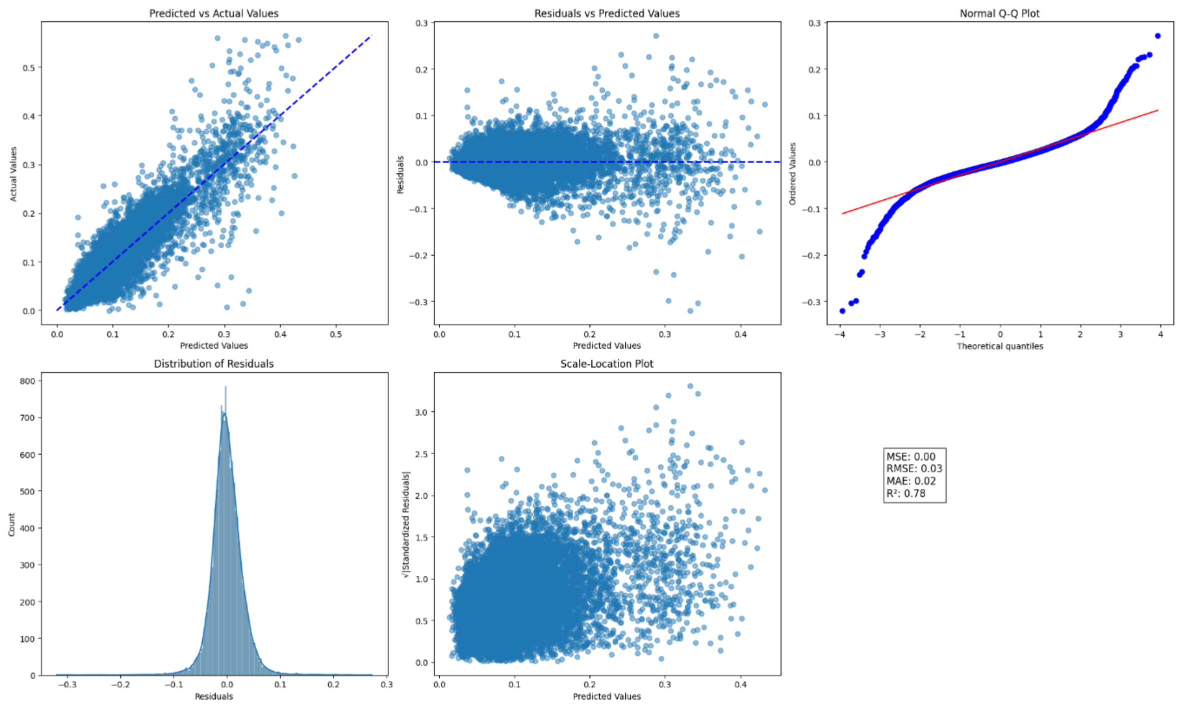


Random Forest Classifier:



Scenario 2:


Residuals Plot
Actual Values
Residuals
Residuals (y_test - predictions)
Actual Values (y_test)


Predicted vs Actual Values
Residuals vs Predicted Values
Normal Q-Q Plot
Distribution of Residuals
Scale-Location Plot
Actual Values
Predicted Values
Residuals
Ordered Values
Theoretical quantiles
Count
√|Standardized Residuals|
MSE: 0.00
RMSE: 0.03
MAE: 0.02
R²: 0.78

Final Project Report

7. Advantages & Disadvantages

Advantages

1. **Automation of Astronomical Tasks**
   The system significantly reduces the time and effort needed to classify galaxies and estimate redshifts, tasks that would otherwise require expert manual intervention.
2. **High Accuracy and Performance**
   With f1-score exceeding 0.862 for classification and R^2 over 0.79 for regression, the models perform exceptionally well on real SDSS data.
3. **Scalable and Extensible**
   The modular architecture allows easy extension to other scenarios like AGN classification or integration of more complex features.
4. **User-Friendly Interface**
   The Flask web application makes the tool accessible to users with no coding background, enabling real-time interaction with machine learning models.
5. **Reproducible and Portable**
   All components (models, scalers, encoders) are saved and versioned, ensuring consistency across experiments and deployments.

Disadvantages

1. **Dependent on Data Quality**
   The model's performance is sensitive to noisy or incomplete SDSS data. Missing values or outliers can degrade prediction accuracy.
2. **Limited to Photometric Features**
   Spectral features, which could enhance performance, were not included due to dataset limitations.
3. **Not Real-Time for Massive Data**
   Though the web app works in real time for individual predictions, batch processing large-scale datasets would require further optimization or backend enhancement.
4. **Scenario 3 Not Implemented**
   The AGN identification scenario was excluded due to the absence of labeled AGN data, limiting the project's completeness.

8. Conclusion

This project successfully demonstrates the use of machine learning techniques in solving key astronomical problems using real-world SDSS data. By implementing robust classification and regression pipelines, the system can classify galaxies by star formation type and estimate their redshift based on photometric features.

The models were optimized through hyperparameter tuning and evaluated rigorously using appropriate metrics. Furthermore, the deployment through a clean Flask-based UI provides a seamless user experience.

Overall, the project bridges data science and astronomy, delivering an intelligent solution that is accurate, scalable, and user-friendly.

9. Future Scope

- **Implement AGN Classification (Scenario 3)**
With the availability of labeled AGN data, the third scenario can be added to enhance the system's scientific utility.

- **Batch Prediction and CSV Upload**
Enable bulk galaxy predictions through file uploads, allowing astronomers to process large datasets efficiently.

- **Integration with Cloud Services**
Deploy the system on cloud platforms (e.g., AWS, Azure) to make it globally accessible and scalable.

- **Incorporate Spectroscopic Features**
Extend the dataset with spectral indices to improve the redshift estimation model's precision.

- **Visualization Dashboard**
Add graphical outputs (e.g., galaxy type distribution, redshift histograms) to provide interactive insights to users.

- **Model Explainability**
Integrate SHAP or LIME to interpret model decisions and build trust with scientific users.