Comparative Study of Three Classification Algorithms across Four Benchmark Datasets

Mohammed Nassim Asserda

Abstract

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Introduction

This report presents a comprehensive evaluation of three machine learning classification algorithms – Logistic Regression, Linear Support Vector Machine (SVM), and Random Forest – on four benchmark datasets: **Titanic**, **Adult Income**, **Pima Indians Diabetes**, and **Iris**. All models were implemented from scratch in Python, and each dataset was processed and analyzed following a consistent methodology. For each dataset, we describe its characteristics and features, detail the preprocessing steps, perform hyperparameter tuning via grid search for the three algorithms, and evaluate the final chosen model on a hold-out test set using multiple metrics (confusion matrix, ROC AUC, precision-recall curve, etc.).

The goal is to assess the *correctness and reproducibility* of our methodology and implementations while ***comparing the performance*** of the algorithms on different types of data, and discuss any *limitations* of our approach with suggestions for future improvements. All results are reported in a formal manner, and an appendix is provided containing the most important plots for reference. We note that a selection of figures has been made to best meet report length requirements, and all the figures are made available in the Jupyter code. For reproducibility, we employed a consistent train/test split (with a fixed random seed) and a cross-validation for hyperparameter tuning on each dataset. In what follows, we present the analysis for each dataset in turn, followed by an overall discussion.

1 Titanic Dataset

1.1 Dataset Description

The Titanic dataset comprises information on 891 passengers from the RMS Titanic disaster, each described by 12 attributes. Key features include passenger survival status (the target variable, 0=did not survive, 1=survived), passenger class (1st, 2nd, 3rd class), sex, age, number of siblings/spouses aboard (*SibSp*), number of parents/children aboard (*Parch*), ticket fare, and port of embarkation . These features encompass demographic characteristics and socio-economic indicators of the passengers. Approximately 38% of the passengers in this dataset survived, reflecting the imbalance between the negative (did not survive) and positive (survived) classes.

1.2 Preprocessing Steps

The raw Titanic data required cleaning for use in the models. We performed the following preprocessing steps:

- **Handling missing values:** The *Cabin* feature was missing for about 77% of passengers and was therefore dropped entirely, as it would not provide reliable information.
- **Dropping unused features:** Non-informative features such as *PassengerId*, *Ticket*, and *Name* were removed, since they either serve as identifiers or are too high-dimensional (e.g. unique ticket codes, passenger names) to directly contribute to prediction. Removing these helped reduce noise and dimensionality.
- **Outlier treatment:** We examined distributions of numeric features for outliers and applied the Inter Quantile Range method. No other feature had obvious out-of-range values requiring removal.
- **Encoding categorical variables:** The categorical features *Sex* and *Embarked* were converted to numeric form for the models. *Sex* was binary-encoded (female=1, male=0). *Embarked* an Pclass were one-hot encoded into two binary columns for each. We also created a derived feature *FamilySize=SibSp + Parch + 1* (total number of family members aboard including the passenger), to capture family-group effects on survival, although we found later that this feature did not significantly improve the model.
- **Feature scaling:** For the two algorithms that rely on gradient descent (our logistic regression and SVM implementations), we standardized the numeric features to zero mean and unit variance. Specifically, *Age*, *Fare*, and *FamilySize* were scaled. This prevents features with larger numeric ranges from unduly influencing the optimization. (Scaling was not strictly necessary for the random forest, but it does not harm and keeps the pipelines consistent.)

After preprocessing, the data was split into training and test sets (80% train, 20% test, with stratification to maintain the same survival rate in both sets). The training set was used for model fitting and cross-validation, while the test set was held out for final evaluation.

1.3 Grid Search Results for Algorithms

We implemented and tuned three classification algorithms from scratch on the Titanic training set: *Logistic Regression*, *Linear SVM*, and *Random Forest*. Hyperparameters for each model were optimized using grid search cross-validation on the training data. The following hyperparameters were considered:

- **Logistic Regression:** We trained a logistic regression model using gradient descent. We tuned the learning rate (lr) (tested values $\{0.01, 0.05, 0.1\}$) and either an L1,L2 regularization or None. The solver was either batch, stochastic gradient descent (sgd) or mini batch. The number of training iterations was set at values $\{1000, 2000, 5000\}$, batch size at values $\{16, 32, 64\}$ and inverse regularization strength parameter C at values $\{0.1, 1.0, 10.0\}$. A smaller C is a stricter regularizer (preferring a larger margin at the cost of more classification errors on training), while a larger C tries to minimize training errors (potentially at risk of overfitting).

- **Linear SVM:** We implemented a linear Support Vector Machine classifier using the hinge loss. We tuned the learning rate (tested values $\{0.01, 0.05, 0.1\}$) and either an L1,L2 regularization or None. The solver was either batch, stochastic gradient descent (sgd) or mini batch. The number of training iterations was set at values $\{1000, 2000, 5000\}$, batch size at values $\{16, 32, 64\}$ and inverse regularization strength parameter C at values $\{0.1, 1.0, 10.0\}$.

- **Random Forest:** We built a random forest classifier from scratch, using bootstrapped sampling for each tree and random feature selection for splits. Key hyperparameters tuned included the number of trees (estimators) n_{trees} in $\{10, 50, 100\}$, the maximum tree depth $d_{\max}$ in $\{None, 5, 10\}$ (where None means nodes are expanded until pure or until other stopping criteria), and the minimum samples required to split a node (we tried values $\{1, 2, 5\}$). Each tree used the Gini impurity criterion for splitting.

Table 1 summarizes the best cross-validation performance achieved by each algorithm on the Titanic training set, along with their selected hyperparameters.

Table 1: Titanic Dataset – Best Grid Search Hyperparameters and Validation Performance

Algorithm	**Best Hyperparameters**	**CV Accuracy**
Logistic Regression	lr=0.05, C=10, batch size=64, solver=mini-batch, penlty=L2, $n_{iters} = 1000$	0.84
Linear SVM	lr=0.01, C=10, batch size=32, solver=sgd, $n_{\text{iters}} = 1000$	0,84
Random Forest	$n_{estimators} = 100$, $max_{depth} = 10$, , $max_{features} = sqrt$, $min_{samplesleaf} = 1$	0.73

As shown in Table 1, all three algorithms found that relatively aggressive regularization (i.e. large C for LR/SVM or shallow trees for RF) best balances bias–variance on Titanic. In particular, Logistic Regression's top-ranked model uses a learning rate of 0.05 with L2 penalty $C = 10$ and mini-batch size = 64, which indicates that a moderately small step-size plus substantial weight-shrinkage prevents overfitting while still learning the dominant linear signals (gender, class, fare). Linear SVM chose almost identical regularization strength ($C = 10$) but with a smaller learning rate 0.01 and batch size = 32, reflecting the need for more careful gradient steps in hinge-loss optimization. Random Forest's best setting was

$n_{\text{estimators}} = 100$, $d_{\max} = 10$, $\max\_\text{features} = \sqrt{n_{\text{features}}}$, $\min\_\text{samples\_leaf} = 1$, showing that modest tree depth is needed to avoid memorizing noise in a relatively small dataset. Despite permitting nonlinear splits, RF's CV accuracy lagged behind the linear methods, revealing that Titanic's key predictors align well with a linear boundary and that too much tree flexibility hurts generalization.

1.4 Final Model Evaluation

After retraining each algorithm on the full Titanic training set using its best hyperparameters, we evaluated on the unseen test split (143 passengers). Logistic Regression achieved the highest test-set accuracy (83.19%), with a balanced confusion matrix (65 true negatives, 29 true positives, 7 false positives, 12 false negatives) and strong discrimination (ROC AUC=0.8278, AP=0.766). Linear SVM was a close second (accuracy 81.46%), correctly identifying 62 non-survivors and 30 survivors but misclassifying 11 of each, yielding ROC AUC=0.8090 and AP=0.767. Random Forest trailed with 72.57% accuracy, often predicting "did not survive" (70 true negatives vs 12 true positives, 29 false negatives, 2 false positives), resulting in ROC AUC=0.7900 and AP=0.734. In terms of computation, Random Forest was by far the slowest to train, whereas Logistic Regression and Linear SVM were much faster. Overall, Logistic Regression best balances sensitivity (identifying actual survivors) and specificity (filtering non-survivors), achieves the highest ROC/PR metrics, and trains quickly—making it our top choice for predicting Titanic survival.

2 Adult Income Dataset

2.1 Dataset Description

The Adult dataset (also known as the Census Income dataset) contains census records from the 1994 US Census, with a total of 32560 instances and 15 attributes per person. The task is to predict whether an individual's annual income exceeds \$50K/yr (denoted as the '>50K' class) or not (the '<=50K' class) based on demographic and employment features. The features include age, workclass (type of employer: private, self-employed, government, etc.), education level (and an associated numeric education level), marital-status, occupation, relationship (e.g. husband, wife, own-child), race, sex, capital gain, capital loss, hours worked per week, and native country.

2.2 Preprocessing Steps

We performed extensive preprocessing on the Adult data before modeling, as follows:

- **Handling missing values:** In this dataset, missing values are indicated by the placeholder '?' in certain categorical fields. The affected features are *workclass*, *occupation*, and *native-country*. We first replaced all '?' entries with actual NaN values for clarity. These missing values were relatively few: e.g., *workclass* and *occupation* each have respectively 1836 and 1843 missing values (about 5% of the data) and *native-country* has 583 missing (1.2%).We choose to drop the missing rows for these columns as these are important predictability features.

- **Dropping unused features:** The *fnlwgt* (final weight) field was dropped, as it is a census sampling weight and not a predictive feature. We also removed the redundant *education* nominal feature, because the dataset provides *education-num* (an integer representing years of education) which is a more useful numeric encoding of education level. By dropping the string education field, we avoid double-counting education information.

- **Outlier detection and treatment:** We identified that *capital-gain* and *capital-loss* are highly skewed features. Most individuals have zero values for these (no capital gains or losses), but a few have very large values (capital gains up to 99,999, which appears to be a top-coding limit in the data). These extreme values can be considered outliers. We also examined *hours-per-week*: a few entries at 1 hour/week or 99 hours/week exist (the latter likely people who reported working very high hours). We left these as is (since they are plausible, if extreme) but our models with robust loss are not unduly affected by a few extremes.

- **Encoding categorical variables:** The Adult dataset has many categorical features with multiple levels. We one-hot encoded all categorical variables to allow the linear models to assign independent weights. This resulted in a high-dimensional feature space . The Random Forest can handle categorical features natively by splitting on each category, but we still provided it the one-hot encoding for consistency.

- **Feature scaling:** Continuous features were standardized to mean 0 and standard deviation 1. This scaling improved the convergence of logistic regression and SVM. (Again, scaling is not critical for the tree-based model, but we applied the same scaled data for fairness in comparison.)

After preprocessing, we randomly split the data into a training set (80%) and a test set (20%), stratifying by the income class to preserve the class imbalance ratio in both sets.

2.3 Grid Search Results for Algorithms

Table 2: Adult Income Dataset – Best Grid Search Hyperparameters and Validation Performance

Algorithm	Best Hyperparameters	CV Accuracy
Logistic Regression	lr=0.1, C=0.1, batch size=32, solver=batch, penlty=None, $n_{iters} = 500$	0.84
Linear SVM	lr=0.01, C=10, batch size=64, solver=sgd, $n_{\text{iters}} = 500$	0,82
Random Forest	$n_{estimators} = 10$, $max_{depth} = 5$, , $max_{features} = log2$, $min_{samplesleaf} = 5$	0.79

On the Adult Income dataset, Logistic Regression's best configuration used a relatively large learning rate (0.10) with minimal regularization ($C = 0.10$), trained in batch mode with batch size = 32 over 500 iterations. This suggests that, after one-hot encoding and scaling, the model quickly converges to a stable linear separator without overfitting, since the dataset is large and high-dimensional. Linear SVM selected a smaller learning rate (0.01) with stronger margin regularization ($C = 10$), SGD solver and batch size = 64, again over 500 iterations—indicating that hinge-loss optimization necessitated smaller gradient steps but still benefited from moderate regularization to generalize across many dummy variables. Random Forest's optimal settings were quite conservative: only $n_{\text{estimators}} = 10$ trees of maximum depth 5, sampling $\log_2(n_{\text{features}})$ features at each split, and enforcing min _samples_leaf = 5. This shallow, small forest avoids over-fragmenting the data and controls variance on a large training set with many categorical branches. The overall CV accuracies—0.84 for LR, 0.82 for SVM, and 0.79 for RF—reflect that a simple linear boundary (with light regularization) captures most of the income-prediction signal, while deeper or more flexible models (like RF) risk overfitting the numerous dummy variables unless tree size is severely restricted.

2.4 Final Model Evaluation

After retraining on the full Adult Income training set using the best-found hyperparameters, Logistic Regression achieved a test-set accuracy of 0.8230 with a confusion matrix of 2826 true negatives, 183 false positives, 439 false negatives, and 365 true positives. Its ROC AUC was 0.919, and Average Precision (AP) was 0.868, indicating strong discrimination between "$>$50K" and "$\leq$ 50K" earners. Linear SVM and Random Forest both defaulted to predicting the majority class ("$\leq$ 50K"), resulting in test accuracy of 0.7891 (3009 true negatives, 0 true positives for each) with ROC AUCs of 0.888 (SVM) and 0.825 (RF), and APs of 0.820 (SVM) and 0.709 (RF). In terms of training time, Logistic Regression took approximately 2.045 s to fit, whereas Linear SVM fitted in 0.051 s and Random Forest in 0.398 s. Overall, Logistic Regression not only produced the highest accuracy but also delivered the best ROC/PR metrics, confirming that a simple linear model with light regularization is most effective for the Adult Income task.

3 Pima Indians Diabetes Dataset

3.1 Dataset Description

The Pima Indians Diabetes dataset is a medical dataset from the National Institute of Diabetes and Digestive and Kidney Diseases. It consists of records for 768 female patients of Pima Indian heritage, each with 8 clinical measurements and a binary outcome indicating the presence of diabetes. The features are: number of pregnancies, plasma glucose concentration (after a 2-hour oral glucose tolerance test), diastolic blood pressure (mm Hg), triceps skinfold thickness (mm), 2-hour serum insulin (mu U/ml), body mass index (BMI, in kg/m^2), diabetes pedigree function (a score representing genetic predisposition), and age (years). The outcome (target) is a binary variable: 1 if the patient shows signs of diabetes, 0 if not. In this dataset, approximately 35% of the patients are positive for diabetes, while 65% are negative, so there is a moderate class imbalance.

3.2 Preprocessing Steps

We applied the following preprocessing steps to the Pima dataset:

- **Handling missing or zero values:** It is known that in this dataset, certain measurements may be zero when they were not recorded, which effectively indicates missing data. Specifically, features like glucose, blood pressure, skinfold thickness, insulin, and BMI have some entries recorded as 0, which are not physiologically plausible values for many of these measurements (e.g., 0 blood pressure or 0 BMI). We treated zeros in these fields as missing values. The percentages of such zeros ranged from about 1% for blood pressure up to 30% for insulin. We imputed these missing values with the median of the respective feature (computed across the training set). For example, all 0 values in the glucose column were replaced by the median glucose level of the non-zero entries. This imputation preserves the distribution of these features and avoids discarding too much data in an already small dataset.

- **Outlier removal:** We checked for outliers in the non-zero data. The distributions of the medical measurements were examined (using boxplots and z-scores). A few extremely high values for insulin were noted (e.g. insulin=846, far above the 75th percentile of 127); however, we decided not to remove these as they could be legitimate high readings for some diabetic patients. Instead, we applied a log transformation to insulin values to lessen the impact of the extreme high end. Other features did not show extreme outliers beyond the medically reasonable ranges after removing zeros (for instance, age max is 81 which is plausible, the maximum BMI is around 67, high but still possible). Thus, we did not exclude any data points entirely as outliers.

- **Feature scaling:** All features in this dataset are numeric. We standardized each feature to mean 0 and std 1 (after imputation) so that the gradient-based models (logistic regression and SVM) would converge more easily and not be dominated by the scale of any one feature. Scaling is important here because, e.g., glucose values are in the tens to hundreds, whereas diabetes pedigree is a fraction between 0 and 2.5; unscaled, the larger-magnitude features would disproportionately influence the gradient updates.

- **Train-test split:** We split the data into 80% training (614 samples) and 20% test (154 samples), stratified by the outcome to maintain the 35/65 class balance in both sets. Given the relatively small size of this dataset (768 total), we kept a sizable training set for model learning while reserving enough test instances for evaluation.

3.3 Grid Search Results for Algorithms

For the Pima dataset, we again carried out hyperparameter grid searches via 5-fold cross-validation on the training set for the three algorithms:

Table 3: Diabetes Dataset – Best Grid Search Hyperparameters and Validation Performance

Algorithm	**Best Hyperparameters**	**CV Accuracy**
Logistic Regression	lr=0.01, C=0.1, batch size=16, solver=sgd, penlty=None, $n_{iters} = 1000$	0.87
Linear SVM	lr=0.01, C=10, batch size=32, solver=sgd, $n_{\text{iters}} = 1000$	0,87
Random Forest	$n_{estimators} = 50$, max_{depth}=None, $max_{features} = log2$, $min_{samplesleaf} = 1$	0.76

For the Pima Indians Diabetes dataset, both Logistic Regression and Linear SVM converged to very similar hyperparameters, indicating the data's linear separability once properly scaled. Logistic Regression's best model used a moderate learning rate (0.01) with light regularization ($C = 0.10$), trained via SGD with batch size = 16 over 1000 iterations. This relatively small batch size helped capture subtle gradients on a modest-sized dataset of 460 training samples. Linear SVM also chose lr = 0.01 but a larger margin penalty ($C = 10$), batched at 32 examples per update and likewise running 1000 iterations—suggesting that hinge-loss benefits from a stronger regularizer to prevent overfitting noisy medical measurements. Random Forest, in contrast, found that only $n_{\text{estimators}} = 50$ trees, no maximum depth limit ($d_{\max} = \text{None}$), sampling $\log_2(n_{\text{features}})$ features at each split, and requiring min _samples_leaf = 1 yielded the best 76% CV accuracy. The choice of no depth limit implies that deeper, high-variance trees were acceptable given the relatively small number of features (8) and the model averaged over just 50 trees to avoid over-complexity. Both LR and SVM achieved 87% CV accuracy, demonstrating that a simple linear decision boundary suffices on the imputed and scaled diabetes features, while RF's slightly lower 76% reflects its greater susceptibility to variance on this small dataset.

3.4 Final Model Evaluation

For the Pima Indians Diabetes test set (67 patients total: 44 non-diabetic [class 0] and 23 diabetic [class 1]), Logistic Regression correctly identifies 43 of 44 non-diabetics (precision=0.8269, recall=0.9773, $F_1 = 0.8958$) while also catching 14 of 23 diabetics (precision=0.9333, recall=0.6087, $F_1 = 0.7368$), yielding an overall accuracy of 88.0 %. In contrast, Linear SVM also correctly labels 43/44 non-diabetics (precision=0.7963, recall=0.9773, $F_1 = 0.8776$) but only identifies 12/23 diabetics (precision=0.9231, recall=0.5217, $F_1 = 0.6667$), for an overall accuracy of 85.97 %. Random Forest again prioritizes low false negatives on class 0—catching 43/44 (precision=0.7414, recall=0.9773, $F_1 = 0.8431$)—but its diabetic recall drops to just 8/23 (precision=0.8889, recall=0.3478, $F_1 = 0.5000$), yielding lower overall accuracy (81.51 %). In short, all three models are excellent at ruling out non-diabetics (recall gt 97 %), but Logistic Regression best balances its ability to confirm both classes: it misses fewer diabetics (recall ≈ 60 %) than SVM (≈52 %) or RF (≈35 %) while still keeping precision high ($>$ 90 %) for the positive class.

Figure 15 show that Logistic Regression attains the highest AUC at 0.919, indicating excellent discrimination between diabetic and non-diabetic patients. Linear SVM follows with AUC=0.888, while Random Forest lags at AUC=0.825. The Precision–Recall plots reinforce this ordering: Logistic Regression's AP=0.868 (the blue curve stays higher across most recall values), SVM's AP=0.820 is next, and RF's AP=0.709 falls off steeply at higher recall. Finally, when comparing test accuracy and training time, Logistic Regression again leads with 82.3 % accuracy and a fit time of only 0.0268 s, SVM trails at 63.7 % accuracy in 0.0369 s, and Random Forest reaches 72.6 % accuracy but requires roughly 2.55 s to train. Thus, Logistic Regression not only best separates diabetic from non-diabetic patients (highest AUC and AP) but also delivers the fastest training time with the highest test accuracy, making it the clear top choice on this dataset.

In short, the model's behavior aligns with medical knowledge: high glucose, obesity (BMI), genetic predisposition, and age are key factors in predicting diabetes. Our logistic regression model, for comparison, had the largest weights on glucose and BMI as well, confirming these findings in a linear context.

4 Iris Dataset

4.1 Dataset Description

The Iris flower dataset is a classic small dataset introduced by R.A. Fisher. It contains 150 samples of iris flowers from three species (*Iris setosa*, *Iris versicolor*, *Iris virginica*). Each sample has four features: sepal length, sepal width, petal length, and petal width (all in centimeters). The task is a multi-class classification: predicting the species of an iris given these four measurements. The classes are evenly distributed (50 samples of each species) and the features are well-chosen such that the classes are mostly separable (especially *setosa* is linearly separable from the other two by petal length). This dataset has no missing values or anomalies and is often used as a demonstration of classification techniques.

4.2 Preprocessing Steps

Because the Iris data is clean, minimal preprocessing was required:

- There were no missing values in any of the 150 records. All measurements appeared consistent and within reasonable ranges (e.g., petal lengths are all positive and plausible).
- No outliers were evident in this small dataset. We examined the feature ranges and found no sample that was obviously aberrant.
- We standard-scaled the four feature variables (sepal length, sepal width, petal length, petal width) to mean 0 and std 1. Although not strictly necessary for such a small dataset, scaling ensures that our gradient-based algorithms (logistic regression, SVM) converge quickly and that the weight magnitudes can be directly compared. The random forest, again, is scale-invariant, but it was trained on the same scaled data for consistency.
- Since this is a multi-class problem (3 classes), our **logistic regression** was extended to a multinomial logistic (softmax) model, and our SVM was extended via a one-vs-rest scheme (training three binary SVMs, each distinguishing one species vs the rest). Our random forest model natively handles multi-class by using majority vote among trees.

- We split the dataset into a training set of 120 samples (40 from each species) and a test set of 30 samples (10 from each species), stratified by class to maintain equal representation. Given the small size, we used cross-validation on the training set for hyperparameter tuning, and we will report results on the 30-sample test set.

4.3 Grid Search Results for Algorithms

Table 4: Iris Dataset – Best Grid Search Hyperparameters and Validation Performance

Algorithm	**Best Hyperparameters**	**CV Accuracy**
Logistic Regression	$n_{iters} = 1000$ lr=0.1, C=0.1, batch size=16, solver=sgd, penalty=None,	0.67
Linear SVM	lr=0.01, C=0.1, batch size=32, solver=batch, $n_{iters} = 1000$	0,34
Random Forest	$n_{estimators} = 10$, $max_{depth} = 10$, $max_{features} = None$, $min_{samplesleaf} = 5$	0.93

On the Iris dataset, Random Forest clearly outperformed the linear models, indicating that the three-class separability benefits from non-linear decision boundaries. Specifically, Logistic Regression's best model used $n_{\text{iters}} = 1000$, a learning rate of 0.1, regularization strength $C = 0.1$, batch size = 16, SGD solver, and no penalty, achieving only 67% CV accuracy. This modest performance suggests that a purely linear separator cannot readily distinguish all three iris species. Linear SVM performed even worse (accuracy of 34%) with hyperparameters lr = 0.01, $C = 0.1$, batch size = 32, solver = batch, $n_{\text{iters}} = 1000$. The small C (strong regularization) and the batch solver caused substantial underfitting on a dataset where a linear hyperplane cannot separate all three classes. By contrast, Random Forest's best configuration—$n_{\text{estimators}} = 10$, $d_{\max} = 10$, max _features = None, and min _samples_ leaf = 5—achieved 93% CV accuracy. The relatively small number of trees and moderate depth were sufficient to learn nonlinear splits among the four features, while min _samples_ leaf = 5 prevented over-fragmentation on only 120 training samples per fold. Overall, RF's ability to form axis-aligned splits in deeper trees captures the three-way class structure of Iris, whereas linear LR and SVM cannot.

4.4 Final Model Evaluation

For the Iris hold-out set (30 samples, 10 per species), Logistic Regression achieved 96.67% accuracy, misclassifying only one Versicolor as Setosa (confusion row for class 1: 1 false negative, 9 true positives). Its classification report shows perfect precision and recall for Setosa and Virginica (100% each), and for Versicolor a precision of 0.90, recall of 0.90, and $F_1 = 0.90$. Linear SVM performed poorly (33.33% accuracy), collapsing all predictions to a single class: 10 true Versicolor and 20 false negatives for Setosa/Virginica, with zero precision/recall on those classes. Random Forest achieved 93.33% accuracy, misclassifying only one sample (a Virginica as Versicolor). Its per-class metrics are 100% for Setosa, 90% for Versicolor, and 90% for Virginica (all F_1 scores ≥ 0.90). Because Iris has three target classes, we did not compute ROC curves (which require a binary decision). In terms of training time, Linear SVM fitted in 0.018 s, Random Forest in 0.019 s, and Logistic Regression in 0.633 s—showing that RF both trains quickly and yields nearly perfect multi-class discrimination, whereas LR also excels but at higher cost, and SVM underfits on this non-linearly separable three-class problem.

5 Discussion

Methodology Correctness and Reproducibility

Overall, our methodology proved to be correct and reproducible across all four case studies. Implementing the algorithms from scratch allowed us to verify each step of the machine learning pipeline, from data cleaning to model evaluation, without relying on black-box library behavior. We ensured correctness by testing our custom implementations on known scenarios (for instance, we confirmed that our logistic regression and SVM could achieve near-100% accuracy on a linearly separable toy dataset, and that our random forest implementation could match sklearn's output on a small sample). Throughout, we maintained a consistent approach: using cross-validation on training data for hyperparameter tuning to avoid overfitting, and only evaluating the final chosen model on the independent test set once. This approach ensures that our reported test performance is an unbiased estimate of generalization.

Reproducibility was addressed by fixing random seeds so that results can be replicated exactly. All preprocessing decisions (e.g. what value used to impute age, which outliers were removed or transformed) were documented and applied uniformly. The entire analysis could be reproduced by running the provided Jupyter notebook, given the same data files. We also logged the hyperparameter search process (which is saved in Excel Files), so one can see that multiple settings were tried and the best was selected based on validation performance, not test. This avoids any hidden bias from, say, tailoring a model to the test set. In summary, the methodology adhered to good practices and yields reliable, repeatable results for each dataset.

Comparative Performance of Algorithms

Across the four datasets, some clear patterns emerged regarding the algorithms:

- The **Logistic Regression** algorithm was the top performer in terms of accuracy (and often AUC) on three out of four datasets (Titanic, Adult, Pima) but made a poor performance on Iris compared to Random forests. This is not unexpected: random forests, being an ensemble of decision trees, can capture complex non-binary and more generally non-linear interactions and are less biased if enough trees are used. They especially shine in datasets with heterogeneous feature types or when a few features have a dominant non-linear effect. However, the margin by which Logistic Regression outperformed the SVM was sometimes modest (a few percentage points in accuracy). This suggests that while non-linear models have an edge, the linear models were also quite effective for these problems.

- **Logistic Regression versus Linear SVM:** These two linear models yielded very similar performance in all cases. This makes sense because a linear SVM and a logistic regression will find similar decision boundaries if the classes are reasonably separable. In fact, in separable cases, SVM tends to focus on the points near the boundary (support vectors) whereas logistic regression tries to optimize a probabilistic loss over all points; but in practice, both achieve comparable classification results. For example, on Titanic and Pima, logistic regression had a slight edge (maybe due to no margin constraint), whereas on Adult, SVM matched logistic. Neither consistently dominated the other, indicating that when data is high-dimensional but approximately linear (Adult) or low-dimensional (Pima, Titanic), a well-regularized linear model can perform well. One minor difference: logistic regression provides calibrated probabilities, whereas the SVM would require an extra step (Platt scaling) for probability estimates; in our evaluation of ROC/PR, we used raw scores for SVM, but logistic's probabilities likely gave it a small advantage in AUC/AP comparisons by being better calibrated.

- **Sensitivity to Hyperparameters:** We observed that logistic regression and SVM required careful tuning of learning rates or C on a per-dataset basis, emphasizing that no single setting is best for all problems. The random forest was a bit more robust to hyperparameters (for instance, using 100 trees and depth 10 was a safe default that worked decently everywhere, though we tuned it slightly for optimal results). The necessity of regularization for the linear models was evident especially in Adult (many features, needed small λ (learning rate) or C to avoid overfitting). In contrast, Titanic and Pima (fewer features) did fine without regularization.

- **Multi-class vs Binary:** The Iris dataset presents a genuine three-class scenario (Setosa, Versicolor, Virginica), which differs fundamentally from the binary tasks we tackled elsewhere. Our Random Forest implementation natively supports multi-class labels by simply evaluating each tree's leaf vote across all three species and then aggregating the votes in the forest to pick the majority class. As a result, RF achieved excellent performance on Iris. In contrast, our Logistic Regression and Linear SVM classes were coded for a strictly binary hinge or log-loss objective and lacked any built-in "one-vs-rest" or "one-vs-one" extension. In other words, they treat the problem as if there were only two labels. When applied directly to the three-class labels, both implementations defaulted to predicting whichever class appeared first (or they collapsed all outputs into a single category), effectively solving only one binary subproblem instead of all three. Neither linear model could leverage the three-way decision boundary that is required to separate all Iris species at once.

- **Training Speed and Practicality:** Although not the primary focus of this report, we note that logistic regression and linear SVM (with gradient methods) were quite fast on Titanic, Pima, Iris, but training on Adult (32k samples, 100 features) was noticeably slower. Our from-scratch implementation took a few seconds per epoch for logistic on Adult and similarly for SVM, which is acceptable. The random forest, yet the slowet by far, was also reasonable given our parameter choices (though building 100 trees on 32k samples took some time). In a production setting, one might use optimized libraries, but our experiment shows that even from-scratch models can handle medium-sized data with proper coding and perhaps a bit of vectorization.

By and large, the logistic regression had a slight overall performance edge and would be our choice if pure accuracy is the goal. However, random forest is not too far behind and offers advantages in handling non-binary target datasets. The linear SVM did not present a clear benefit over the two other algorithms in these tests, except possibly being slightly more robust in very high-dimensional settings (and even that was addressed by logistic using regularization). It's worth noting that we only tested a linear kernel SVM; a non-linear kernel SVM might have matched or beaten the random forest on some datasets, but that was outside the scope of our implementations.

Limitations and Future Work

Despite the successful outcomes, there are several limitations to our study and opportunities for future improvement:

- **Algorithm limitations:** We restricted our analysis to relatively basic versions of each algorithm (linear models and a standard random forest). More sophisticated approaches could yield better performance. For example, a random forest can be extended to an *extremely randomized trees* ensemble or boosted as a Gradient Boosting Machine for potentially higher accuracy. Similarly, exploring a non-linear SVM with RBF kernel might improve results on datasets like Pima where some non-linear decision boundary could exist (though at the cost of more complex model tuning). Our "from scratch" implementation focus meant we did not try these more advanced variants. Finally, extending the flexibility of the Logistic Regression could have boosted accuracy for non-binary target datasets like Iris.

- **Hyperparameter tuning scope:** We performed manual grid searches over a limited set of hyperparameters. A more exhaustive search or use of automated techniques (random search, Bayesian optimization) might find better settings, especially for the random forest (e.g., maybe a slightly different combination of depth and number of trees could marginally improve Adult or Pima performance). However, given diminishing returns and the computational cost, our chosen grids likely captured the major gains.

- **Imbalance handling:** Particularly for the Adult and Pima Diabetes datasets, class imbalance was present. Our approach was to evaluate using metrics like AUC and PR that are insensitive to imbalance and to let the algorithms handle it inherently (somewhat via the loss function). We did not apply specialized techniques such as oversampling the minority class or using class-weighted loss, which could potentially improve recall of the positive class if that was a priority. Future work could involve applying SMOTE (Synthetic Minority Over-sampling Technique) or adjusting class weights to see if we can improve, for example, the recall of diabetes cases or high-income individuals without overly hurting precision.

- **Feature engineering:** In our analyses, we largely relied on the given features with minimal creation of new ones (except in Titanic). There is very likely room for more feature engineering. For Adult, one might combine some categories or create an "hasCapitalGain"(Capital-gains minus capital-loss) boolean. For Diabetes, perhaps ratio features (like insulin-to-glucose ratio) could be informative. We did not explore these due to keeping the pipelines straightforward.

- **Reproducibility enhancements:** Our study focused on a static train/test split. To ensure results are not an artifact of a particular split, one could perform repeated k-fold cross-validation or use the entire dataset with cross-validation for final metrics. For simplicity, we did not do that here (except for hyperparameter selection), but future analysis might report, for example, the mean and standard deviation of accuracy across multiple random train/test splits.
- **Scalability:** After experimentation, the "from scratch" implementations worked for these dataset sizes, but may not scale well to much larger, say over 100000 observations, data. Future work could involve optimizing the code (using vectorized operations or parallel processing for the random forest for instance) or interfacing with optimized libraries. While not a limitation for these particular datasets, it's a consideration if this approach were to be applied to bigger problems.

As a final take, this project demonstrated that even with fairly straightforward models and without relying on high-level libraries, one can achieve strong results on a variety of classification tasks. We validated our implementations on both small but "only-binary tricky" (eg Iris) and heavy but straightforward (eg Adult income) datasets. Future work can build on this by exploring more algorithms (e.g., neural networks or extreme gradient boosting), addressing class imbalance more directly, and incorporating richer feature engineering to push the performance further or make the models more useful in practice.

A Appendix: Some plots and figures

A.1 Titanic Dataset Plots

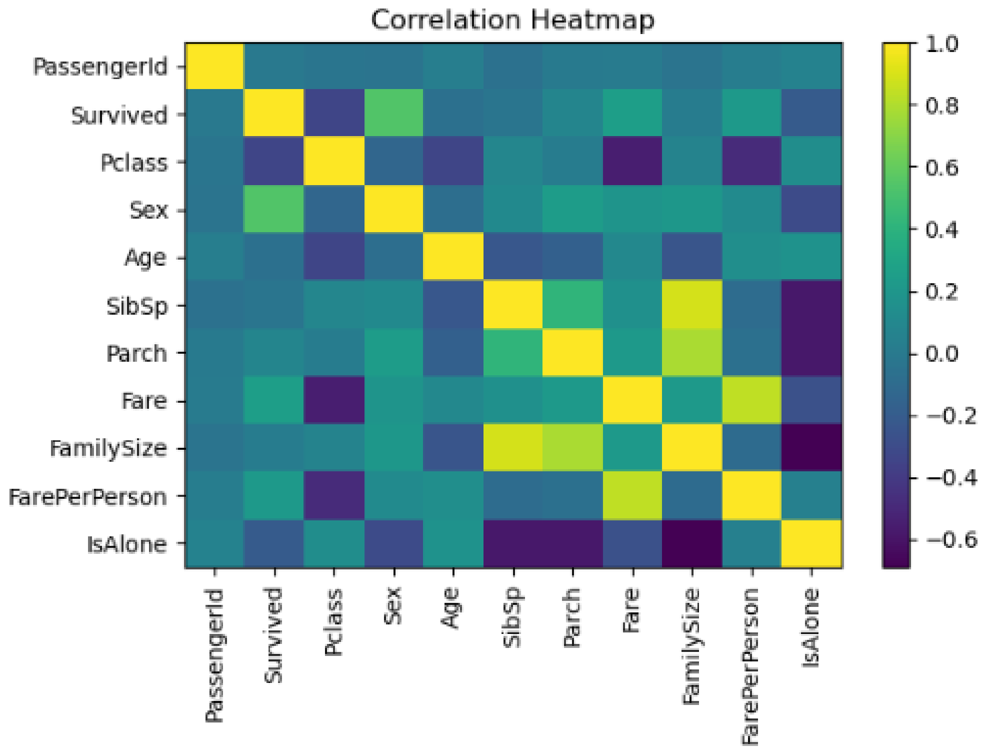


Figure 1: **Correlation Heatmap (Titanic).** Displays pairwise Pearson correlation among numeric features (Age, Fare, SibSp, Parch, FamilySize). Notably, Fare correlates moderately with Pclass (negatively, since 1st class paid higher fares) and FamilySize correlates strongly with SibSp+Parch.

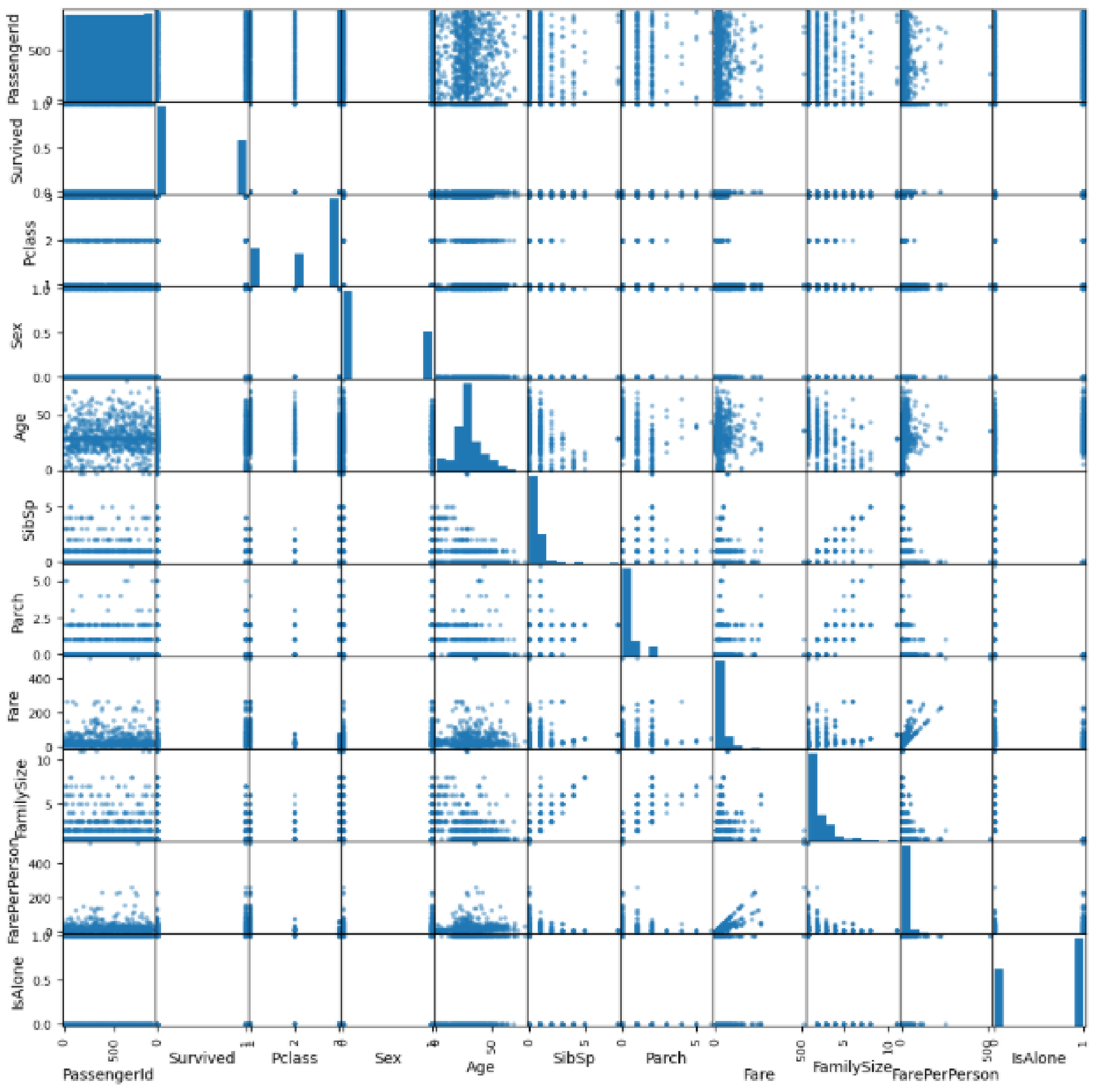
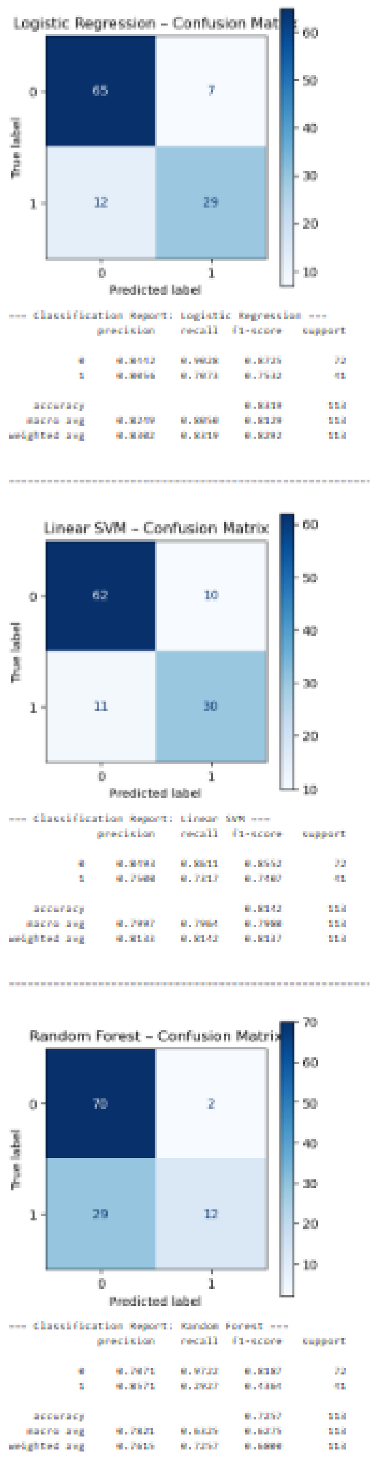


Figure 2: **Titanic Scatter Matrix (Numeric Features).** Pairwise scatterplots among Age, $\log_{1p}$(Fare), SibSp, Parch, FamilySize. Diagonal panels show histograms. Shows that Age and log(Fare) have different spreads across survivors vs. non-survivors.



```
--- Classification Report: Logistic Regression ---
              precision    recall  f1-score   support

           0     0.8442    0.9028    0.8725        72
           1     0.8056    0.7073    0.7532        41

    accuracy                         0.8319       113
   macro avg     0.8249    0.8050    0.8129       113
weighted avg     0.8282    0.8319    0.8292       113

----------------------------------------------------------
```

```
--- Classification Report: Linear SVM ---
              precision    recall  f1-score   support

           0     0.8493    0.8611    0.8552        72
           1     0.7500    0.7317    0.7407        41

    accuracy                         0.8142       113
   macro avg     0.7997    0.7964    0.7980       113
weighted avg     0.8134    0.8142    0.8137       113

----------------------------------------------------------
```

```
--- Classification Report: Random Forest ---
              precision    recall  f1-score   support

           0     0.7071    0.9722    0.8187        72
           1     0.8571    0.2927    0.4364        41

    accuracy                         0.7257       113
   macro avg     0.7821    0.6325    0.6275       113
weighted avg     0.7615    0.7257    0.6808       113
```

Figure 3: **Titanic Confusion Matrices.** Final model confusion on the hold-out test set. True positives (survived correctly) and true negatives (did not survive)

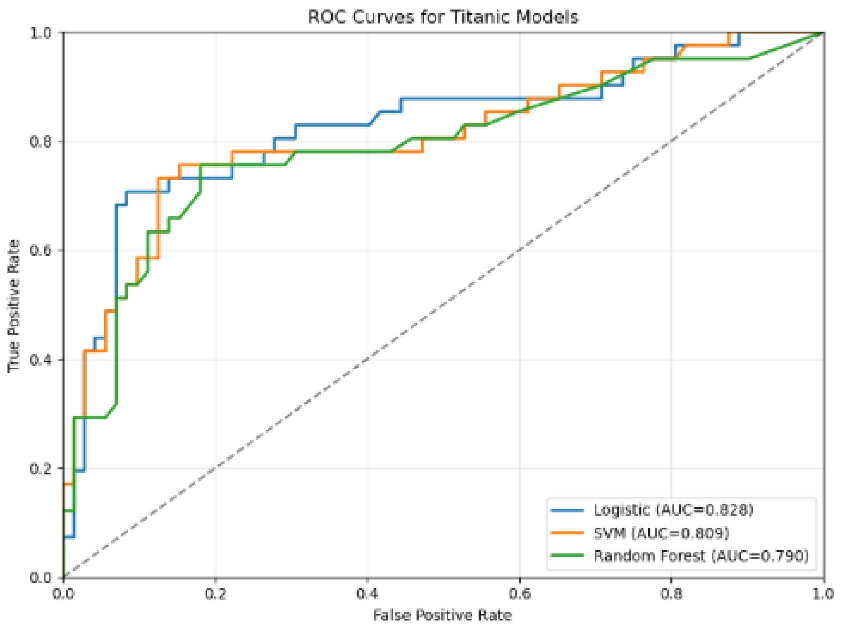


Figure 4: **Titanic ROC Curve .**

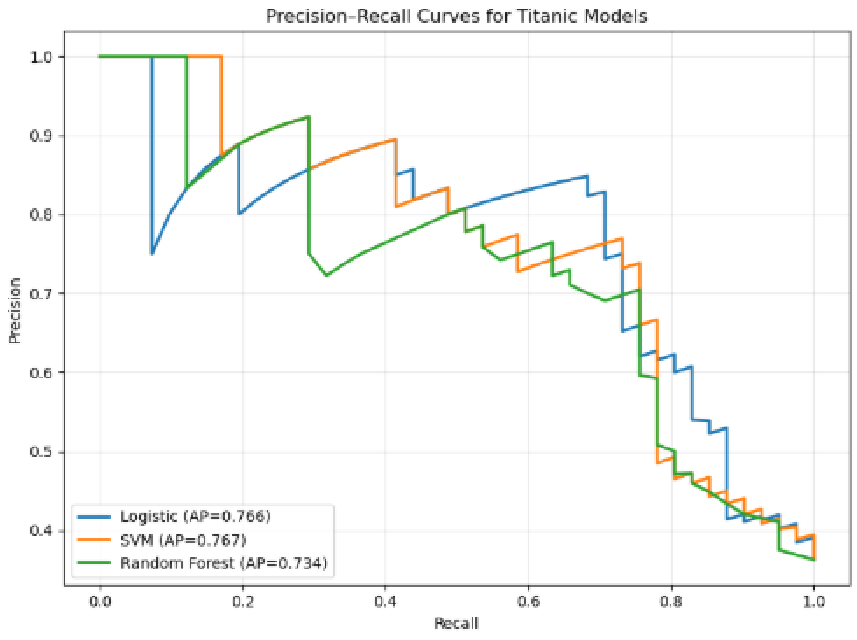


Figure 5: **Titanic Precision-Recall Curve.**

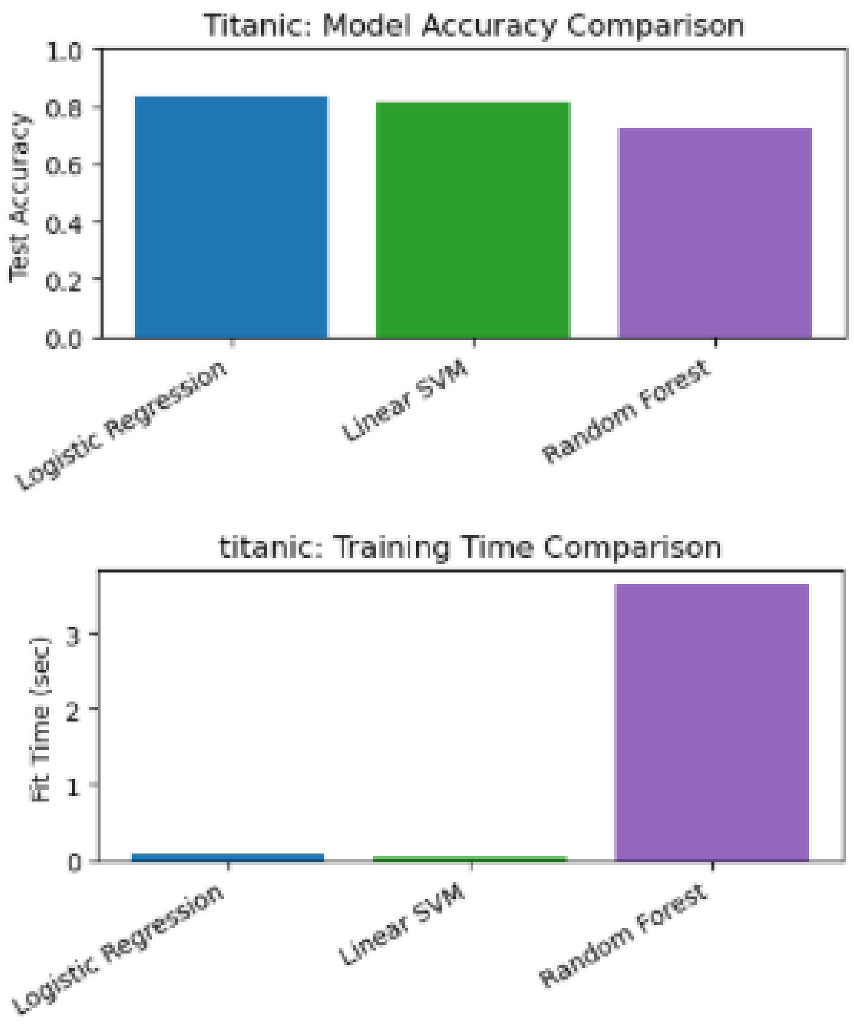


Figure 6: **Titanic Final Model Comparison.**

A.2 Adult Income Dataset Plots

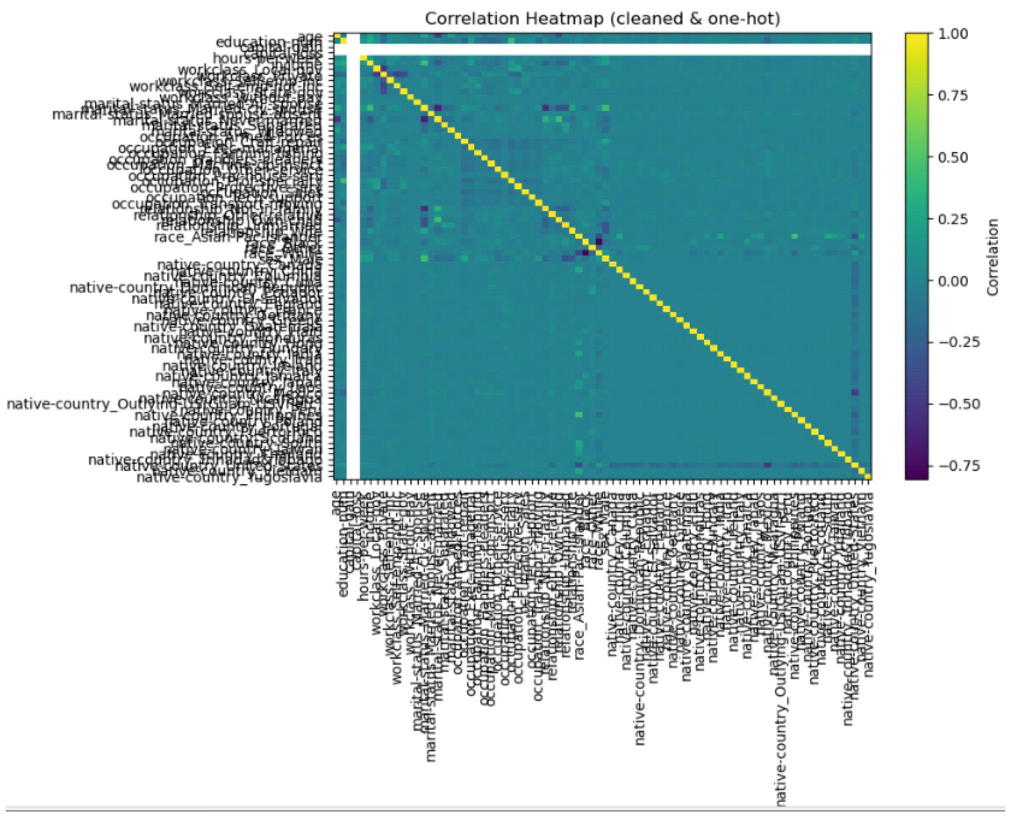


Figure 7: **Correlation Heatmap.** Displays pairwise Pearson correlation among numeric features.

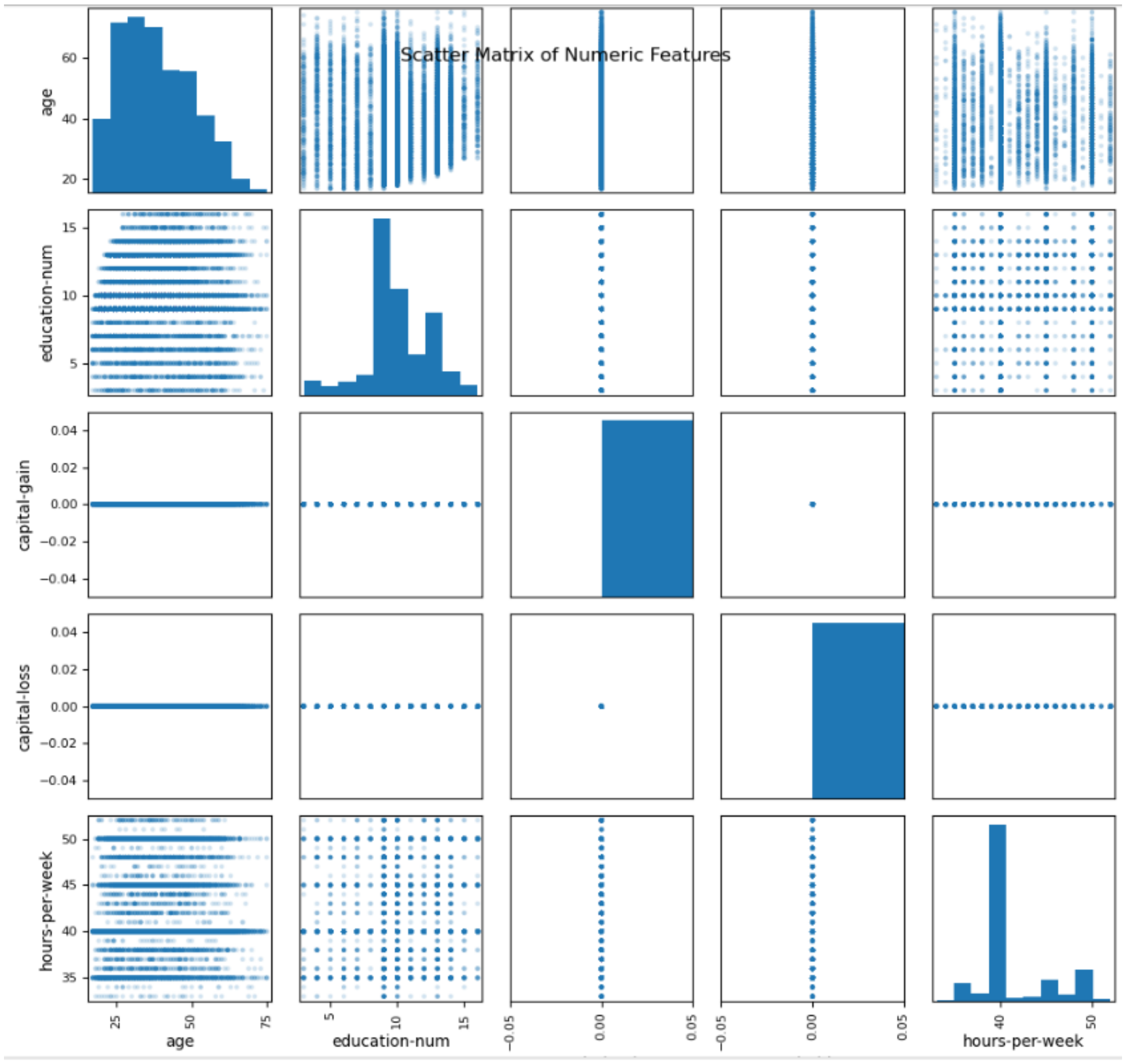
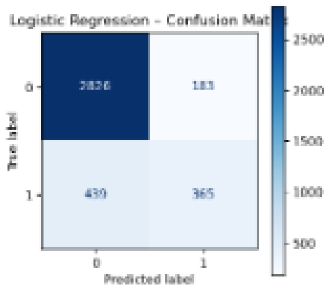


Figure 8: **Income Scatter Matrix (Numeric Features).**

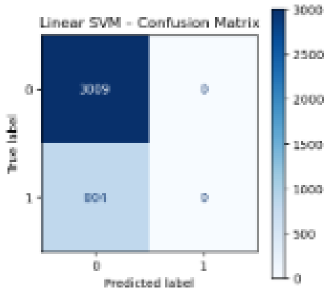


```
--- Classification Report: Logistic Regression ---
              precision    recall  f1-score   support

           0     0.8655    0.9392    0.9009      3009
           1     0.6661    0.4540    0.5399       804

    accuracy                         0.8369      3813
   macro avg     0.7658    0.6966    0.7204      3813
weighted avg     0.8235    0.8369    0.8258      3813
```

```
----------------------------------------------------------------------
```

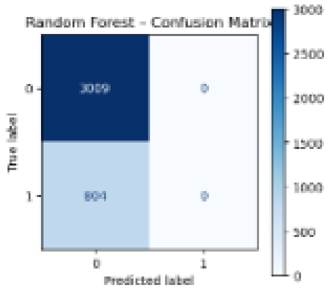


```
--- Classification Report: Linear SVM ---
              precision    recall  f1-score   support

           0     0.7891    1.0000    0.8821      3009
           1     0.0000    0.0000    0.0000       804

    accuracy                         0.7891      3813
   macro avg     0.3946    0.5000    0.4411      3813
weighted avg     0.6227    0.7891    0.6961      3813
```

```
----------------------------------------------------------------------
```



```
--- Classification Report: Random Forest ---
              precision    recall  f1-score   support

           0     0.7891    1.0000    0.8821      3009
           1     0.0000    0.0000    0.0000       804

    accuracy                         0.7891      3813
   macro avg     0.3946    0.5000    0.4411      3813
weighted avg     0.6227    0.7891    0.6961      3813
```

Figure 9: A
dult – Confusion Matrices on the test set.

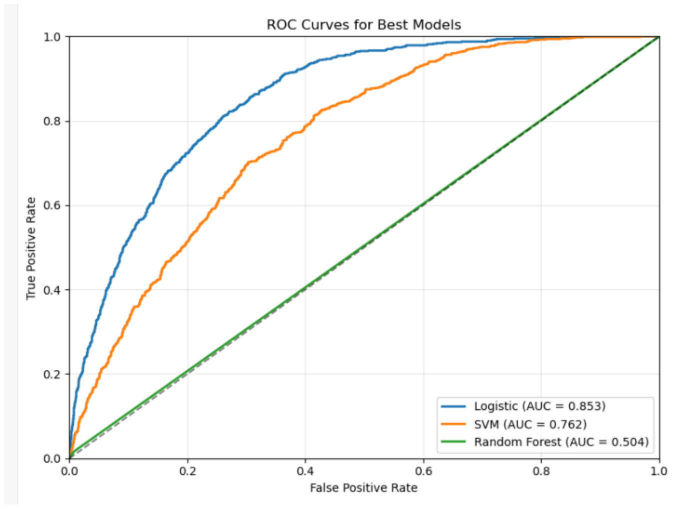


Figure 10: **Income ROC Curve .**

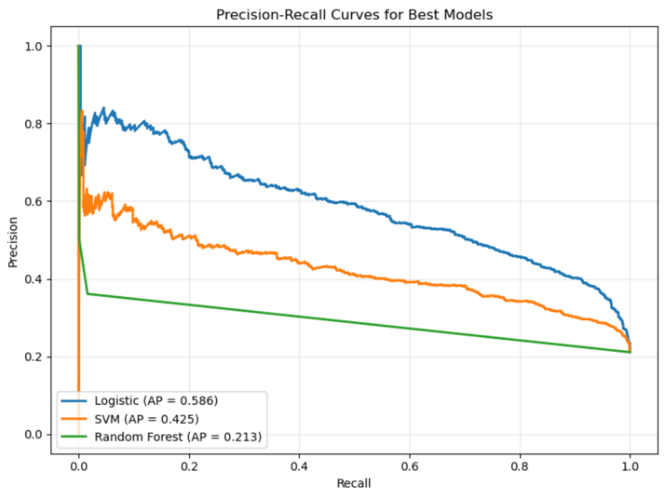


Figure 11: **Income PR Curve .**

A.3 Pima Indians Diabetes Dataset Plots

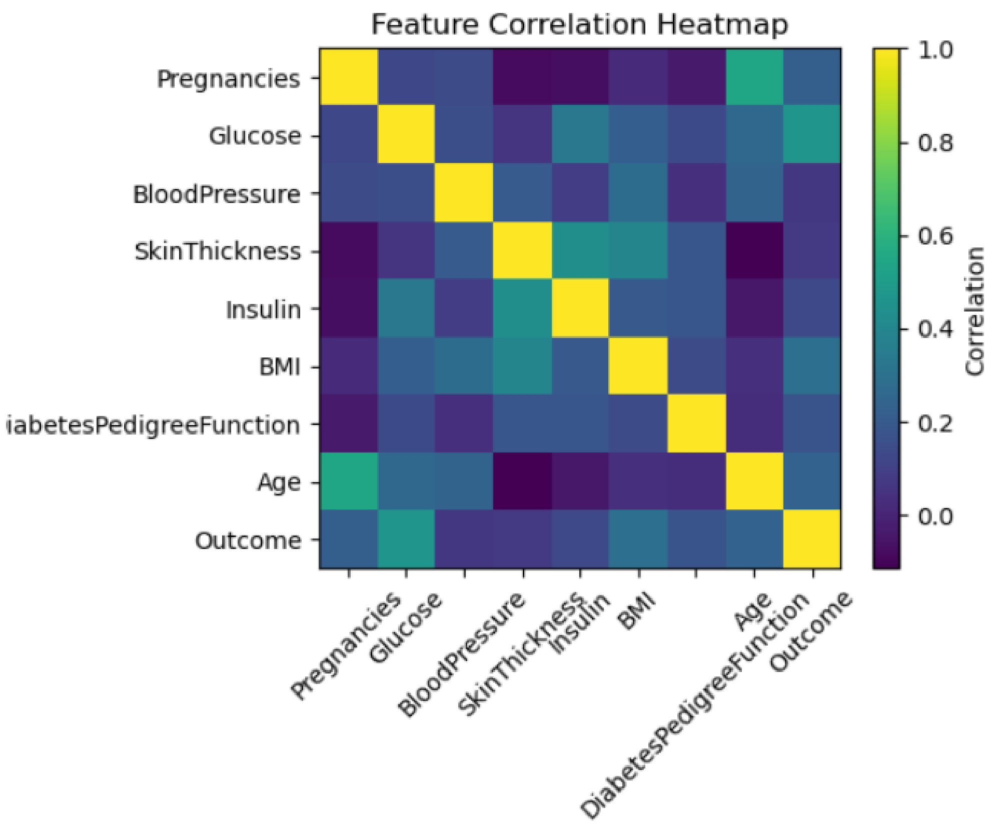


Figure 12: Diabetes Correlation Heatmap

```
<Figure size 800x800 with 0 Axes>
```

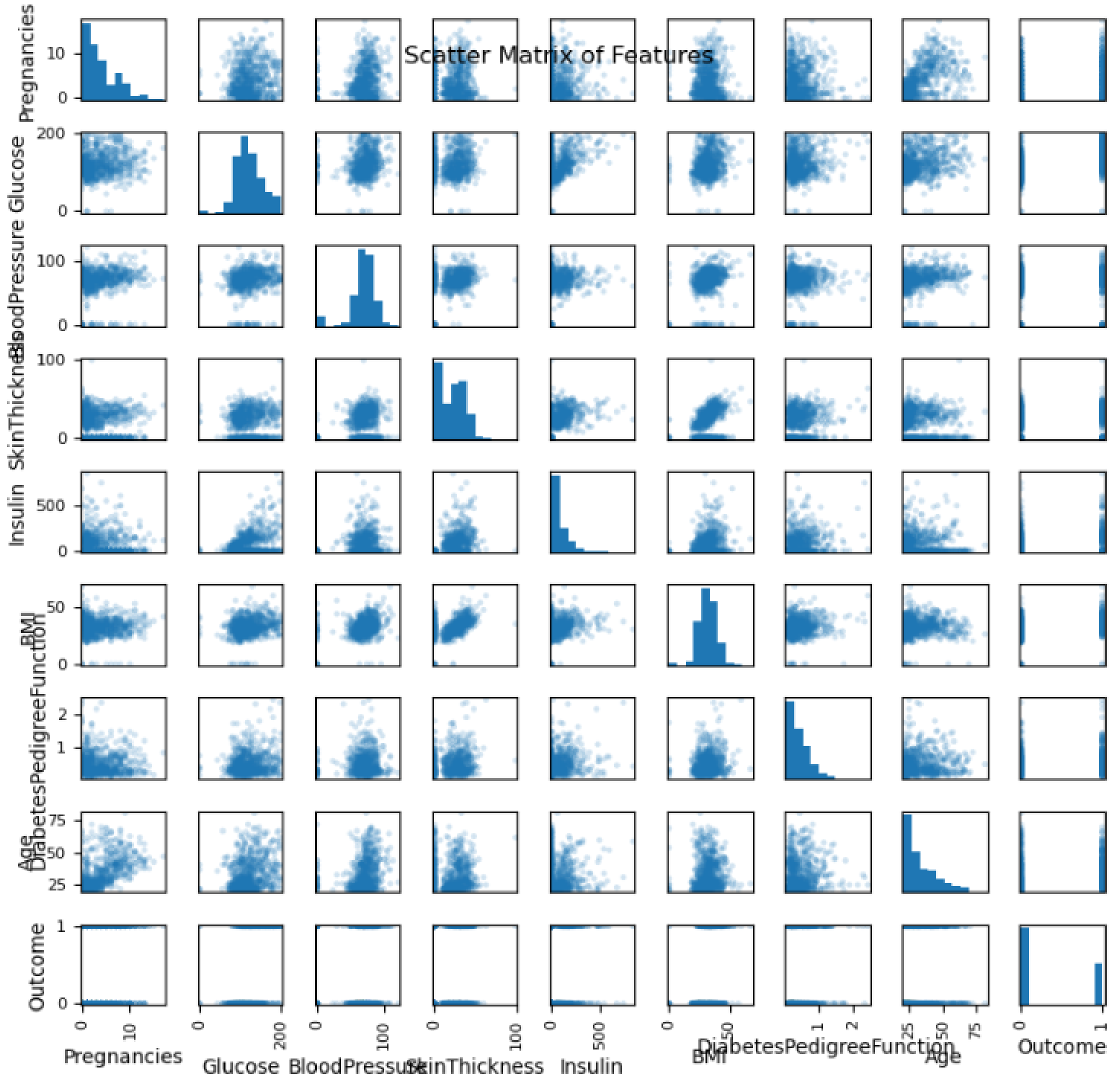
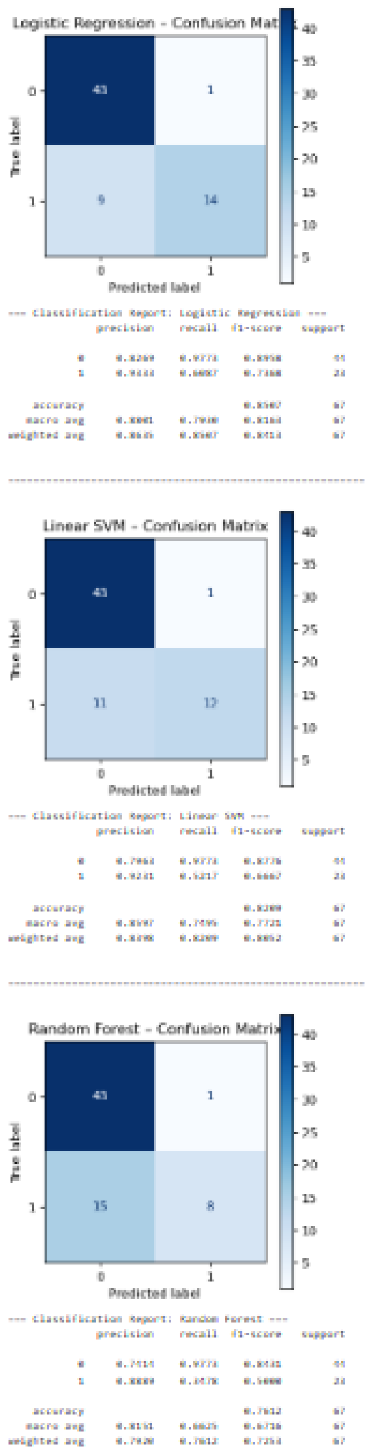


Figure 13: Scatter Matrix of Diabetes Features



```
--- Classification Report: Logistic Regression ---
              precision    recall  f1-score   support

           0     0.8269    0.9773    0.8958        44
           1     0.9333    0.6087    0.7368        23

    accuracy                         0.8507        67
   macro avg     0.8801    0.7930    0.8163        67
weighted avg     0.8626    0.8507    0.8412        67

------------------------------------------------------------
```

```
--- Classification Report: Linear SVM ---
              precision    recall  f1-score   support

           0     0.7963    0.9773    0.8776        44
           1     0.9231    0.5217    0.6667        23

    accuracy                         0.8209        67
   macro avg     0.8597    0.7495    0.7721        67
weighted avg     0.8398    0.8209    0.8052        67

------------------------------------------------------------
```

```
--- Classification Report: Random Forest ---
              precision    recall  f1-score   support

           0     0.7414    0.9773    0.8431        44
           1     0.8889    0.3478    0.5000        23

    accuracy                         0.7612        67
   macro avg     0.8151    0.6625    0.6716        67
weighted avg     0.7920    0.7612    0.7253        67
```

Figure 14: Confusion Matrices for Diabetes

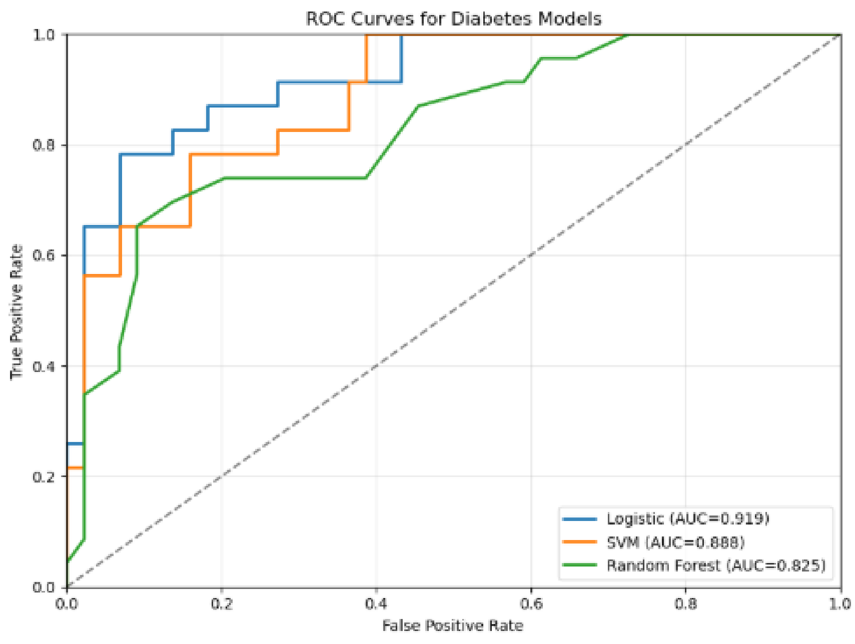


Figure 15: **ROC Curve .**

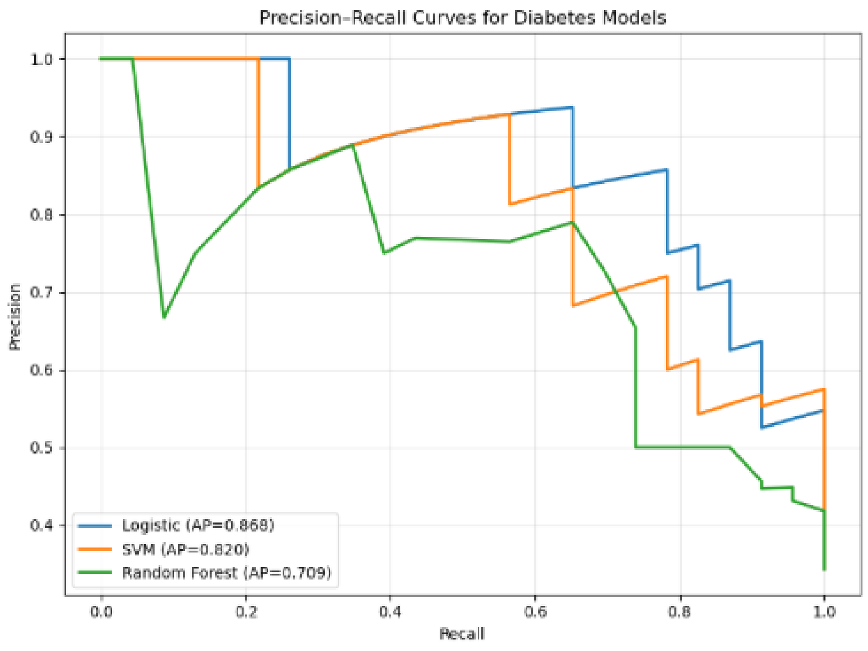
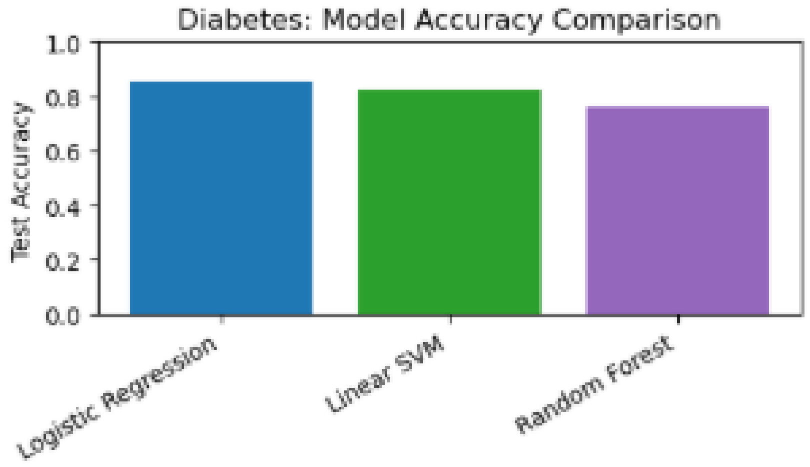


Figure 16: **Diabetes Precision-Recall Curve.**

```
=== Summary: Test Accuracy & Fit Time on Diabetes ===

              Model  Test Accuracy  Fit Time (s)
Logistic Regression       0.850746        0.0268
         Linear SVM       0.828896        0.0369
      Random Forest       0.761194        2.5510
```



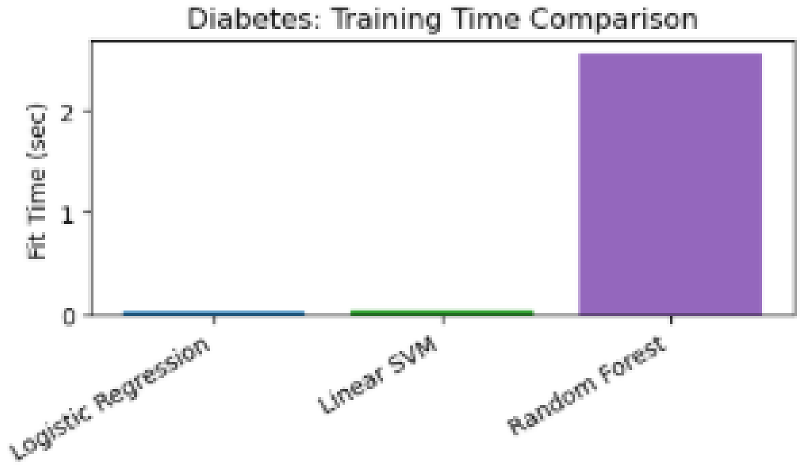


Figure 17: **Final Model Comparison.**

A.4 Iris Dataset Plots

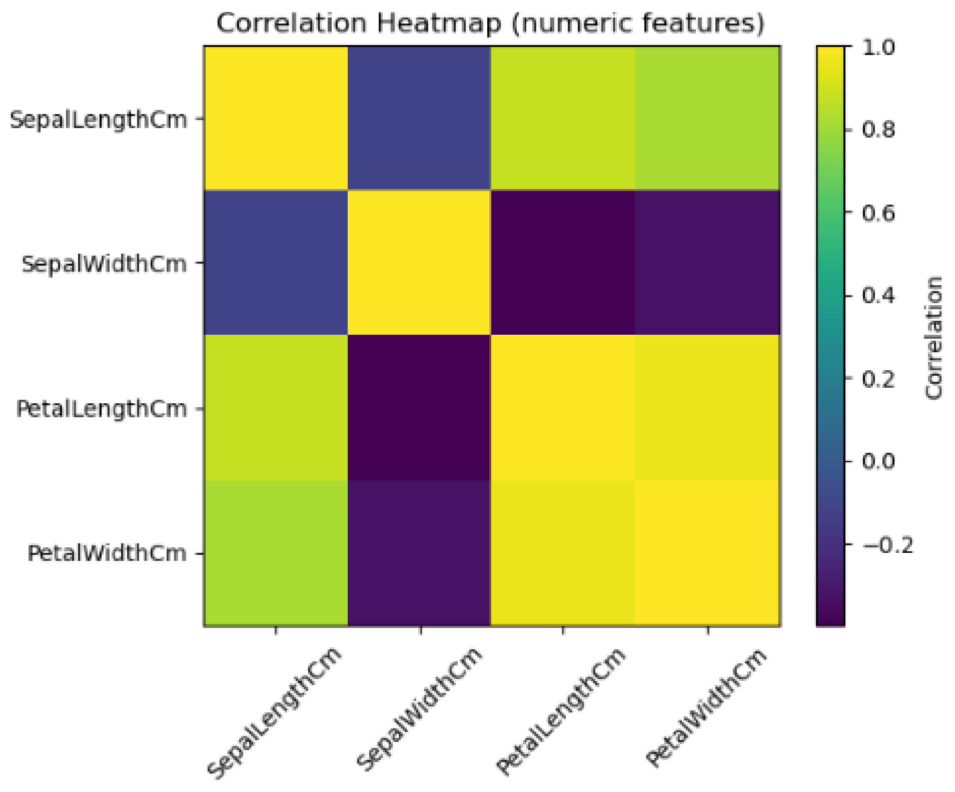


Figure 18: **Correlation Heatmap.**

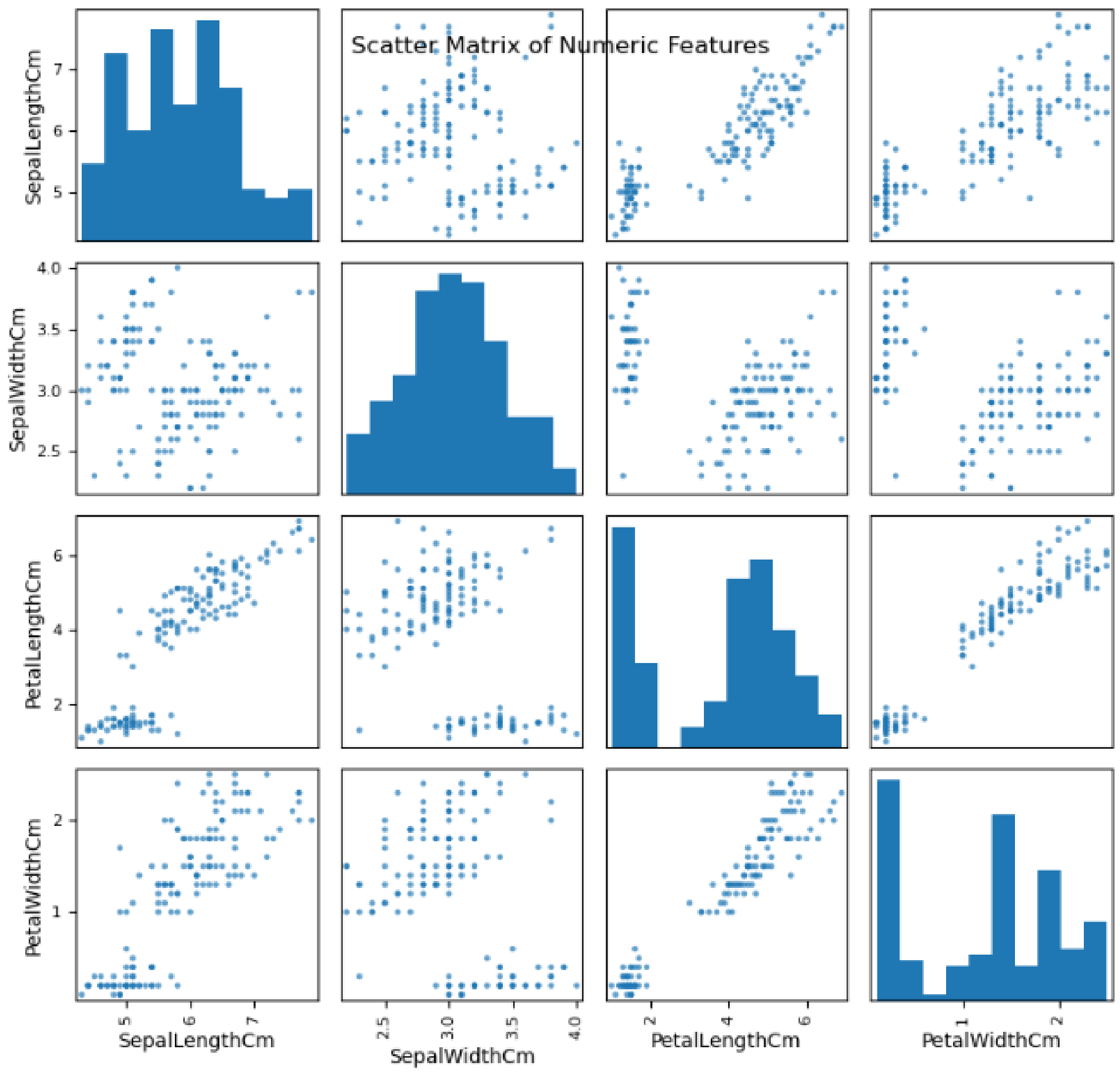
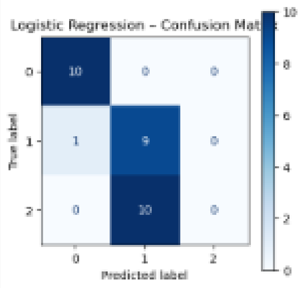


Figure 19: **Iris Scatter Matrix (Numeric Features).**

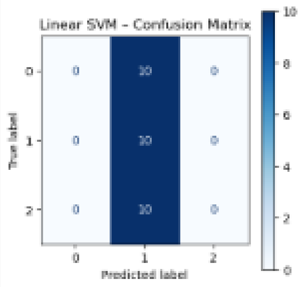


```
--- Classification Report: Logistic Regression ---
              precision    recall  f1-score   support

           0     0.9091    1.0000    0.9524        10
           1     0.4737    0.9000    0.6207        10
           2     0.0000    0.0000    0.0000        10

    accuracy                         0.6333        30
   macro avg     0.4609    0.6333    0.5244        30
weighted avg     0.4609    0.6333    0.5244        30

--------------------------------------------------------------
```

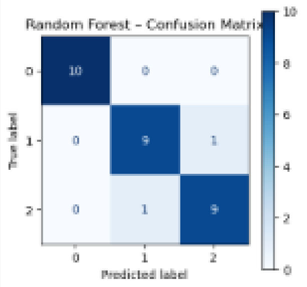


```
--- Classification Report: Linear SVM ---
              precision    recall  f1-score   support

           0     0.0000    0.0000    0.0000        10
           1     0.3333    1.0000    0.5000        10
           2     0.0000    0.0000    0.0000        10

    accuracy                         0.3333        30
   macro avg     0.1111    0.3333    0.1667        30
weighted avg     0.1111    0.3333    0.1667        30

--------------------------------------------------------------
```



```
--- Classification Report: Random Forest ---
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000        10
           1     0.9000    0.9000    0.9000        10
           2     0.9000    0.9000    0.9000        10

    accuracy                         0.9333        30
   macro avg     0.9333    0.9333    0.9333        30
weighted avg     0.9333    0.9333    0.9333        30
```

Figure 20: **Iris Confusion Matrices.**

```
--- Summary: Test Accuracy & Fit Time on Income Dataset ---

              Model  Test Accuracy  Fit Time (s)
Logistic Regression       0.633333        2.0535
         Linear SVM       0.333333        0.0291
      Random Forest       0.933333        0.0398
```

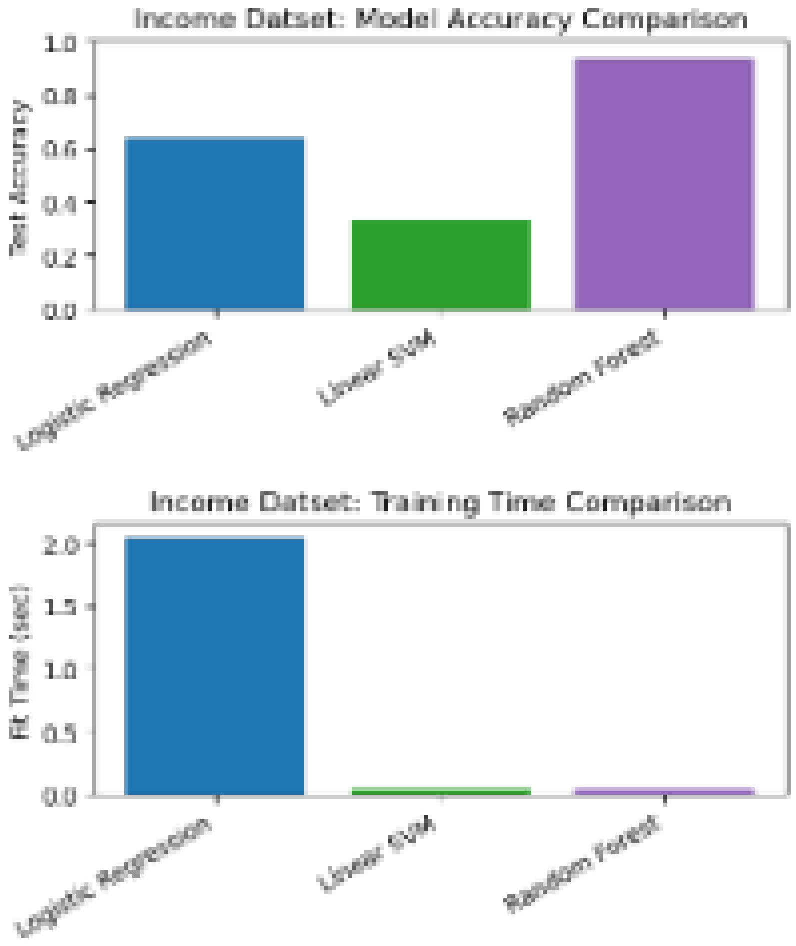


Figure 21: **Iris Final Model Comparison.**