Sri Sivasubramaniya Nadar College of Engineering, Chennai

(Autonomous Institution under Anna University)

Degree & Branch	5 years Integrated M.Tech CSE	**Semester**	V
Subject Code & Name	ICS1512 – Machine Learning Algorithms Laboratory		
Academic Year	2025–2026 (Odd Semester)	**Batch**	2023–2028
Name	Pravin G	**Reg No**	3122237001041

Experiment # 4: Ensemble Prediction and Decision Tree Model Evaluation

Aim:

To build classifiers such as Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Models (using SVM, Naive Bayes, Decision Tree) and evaluate their performance through 5-Fold Cross-Validation and hyperparameter tuning.

Libraries used:

- Numpy
- Pandas
- Scipy
- Scikit-Learn
- Matplotlib.pyplot

Description of the objective performed

- **Data Preparation:** Loaded dataset using kagglehub.dataset download() and converted it into a Pandas DataFrame.
- **Exploratory Data Analysis (EDA):**
  - Performed Numerical Column analysis using histogram and pdf
  - Performed Categorical column analysis using One way ANOVA test
  - Visualized Missing Values
  - Visualized distributions and relationships using:
    - plt.hist() for histograms
    - plt.scatter() for 2D scatter plots
    - sns.heatmap() for feature correlation matrix

- **Data Preprocessing :**
  - Handled Missing Values
  - Outlier Treatment.
  - Encoding categorical column values
  - Standardize
- **Modeling**
  - K-Fold cross validation
  - Model Fitting
- **Evaluation and Visualization**
  - Metrics Accuracy, F1 Score
  - Visualization Confusion Matrix, ROC/AUC Curve

Code :

Train Test Split

```
X_temp, X_test, y_temp, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
X_train, X_val, y_train, y_val = train_test_split(
    X_temp, y_temp, test_size=0.125, stratify=y_temp, random_state=42
)
# Now train=70%, val=10%, test=20%
print("\nTrain size:", X_train.shape, "Val size:", X_val.shape, "Test size:", X_test.shape)
```

Pre-Processing

```
num_features = X.select_dtypes(include=["int64", "float64"]).columns
cat_features = X.select_dtypes(include=["object"]).columns

preprocessor = ColumnTransformer([
    ("num", Pipeline([
        ("imputer", SimpleImputer(strategy="mean")),
        ("scaler", StandardScaler())
    ]), num_features),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_features)
])
```

K-Fold Cross Validation

```
cv = KFold(n_splits=5, shuffle=True, random_state=42)

cv_results = cross_validate(
    best_model, X_train, y_train,
    cv=cv,
    scoring=["accuracy", "f1_weighted"],
    n_jobs=-1,
    return_train_score=False
)

print("\n--- 5-Fold Cross Validation on Best Model ---")
for i, (acc, f1) in enumerate(zip(cv_results["test_accuracy"], cv_results["test_f1_weighted"])
    print(f"Fold {i}: Accuracy = {acc:.4f}, F1 = {f1:.4f}")

print("\nMean Accuracy:", cv_results["test_accuracy"].mean())
print("Mean F1:", cv_results["test_f1_weighted"].mean())
```

Hyperparameter Tuning Tables

criterion	max_depth	Accuracy	F1 Score
entropy	5	0.9547	0.9546
entropy	5	0.9522	0.9521
entropy	None	0.9472	0.9473
entropy	None	0.9471	0.9473
entropy	10	0.9446	0.9447

Table 1: Decision Tree - Hyperparameter Tuning

learning_rate	n_estimators	Accuracy	F1 Score
0.1934	253	0.9774	0.9773
0.3845	142	0.9724	0.9721
0.3437	153	0.9699	0.9697
0.4558	264	0.9699	0.9696
0.3763	239	0.9698	0.9697

Table 2: AdaBoost - Hyperparameter Tuning

n_estimators	learning_rate	max_depth	Accuracy	F1 Score
200	0.2	3	0.9699	0.9697
356	0.1877	34	0.9698	0.9697
363	0.0650	5	0.9698	0.9697
300	0.1	5	0.9673	0.9672
300	0.2	3	0.9673	0.9671

Table 3: Gradient Boosting - Hyperparameter Tuning

n_estimators	learning_rate	max_depth	gamma	Accuracy	F1 Score
300	0.1	7	0.3	0.9748	0.9747
100	0.1	7	0.3	0.9748	0.9747
300	0.1	5	0.3	0.9748	0.9747
200	0.1	5	0.3	0.9748	0.9747
100	0.1	5	0.3	0.9748	0.9747

Table 4: XGBoost - Hyperparameter Tuning

n_estimators	max_depth	criterion	Accuracy	F1 Score
100	20	entropy	0.9573	0.9570
180	26	entropy	0.9573	0.9570
200	10	entropy	0.9573	0.9570
300	5	entropy	0.9573	0.9570
200	None	entropy	0.9573	0.9570

Table 5: Random Forest - Hyperparameter Tuning

Base Models	Final Estimator	Accuracy / F1 Score
SVM, Na¨ıve Bayes, Decision Tree	Logistic Regression	0.9736 / 0.9683
SVM, Na¨ıve Bayes, Decision Tree	Random Forest	0.9666 / 0.9717
SVM, Decision Tree, KNN	Logistic Regression	0.9806 / 0.9752

Table 6: Stacked Ensemble - Hyperparameter Tuning

Result Tables:

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average Accuracy
Decision Tree	0.9375	0.9375	0.9625	0.8987	0.8987	0.9269
AdaBoost	0.9750	0.9625	1	0.9747	0.9367	0.9697
Gradient Boosting	0.9625	0.9625	0.9875	0.9241	0.9367	0.9546
XGBoost	0.9625	0.9750	1	0.9494	0.9241	0.9621
Random Forest	0.9250	0.9500	0.9875	0.9367	0.9367	0.9471
Stacked Model	0.9875	0.95	0.9875	0.9494	0.9620	0.9672

Table 7: 5-Fold Cross Validation Results for All Models

Visualization

Decision Tree

Confusion Matrix

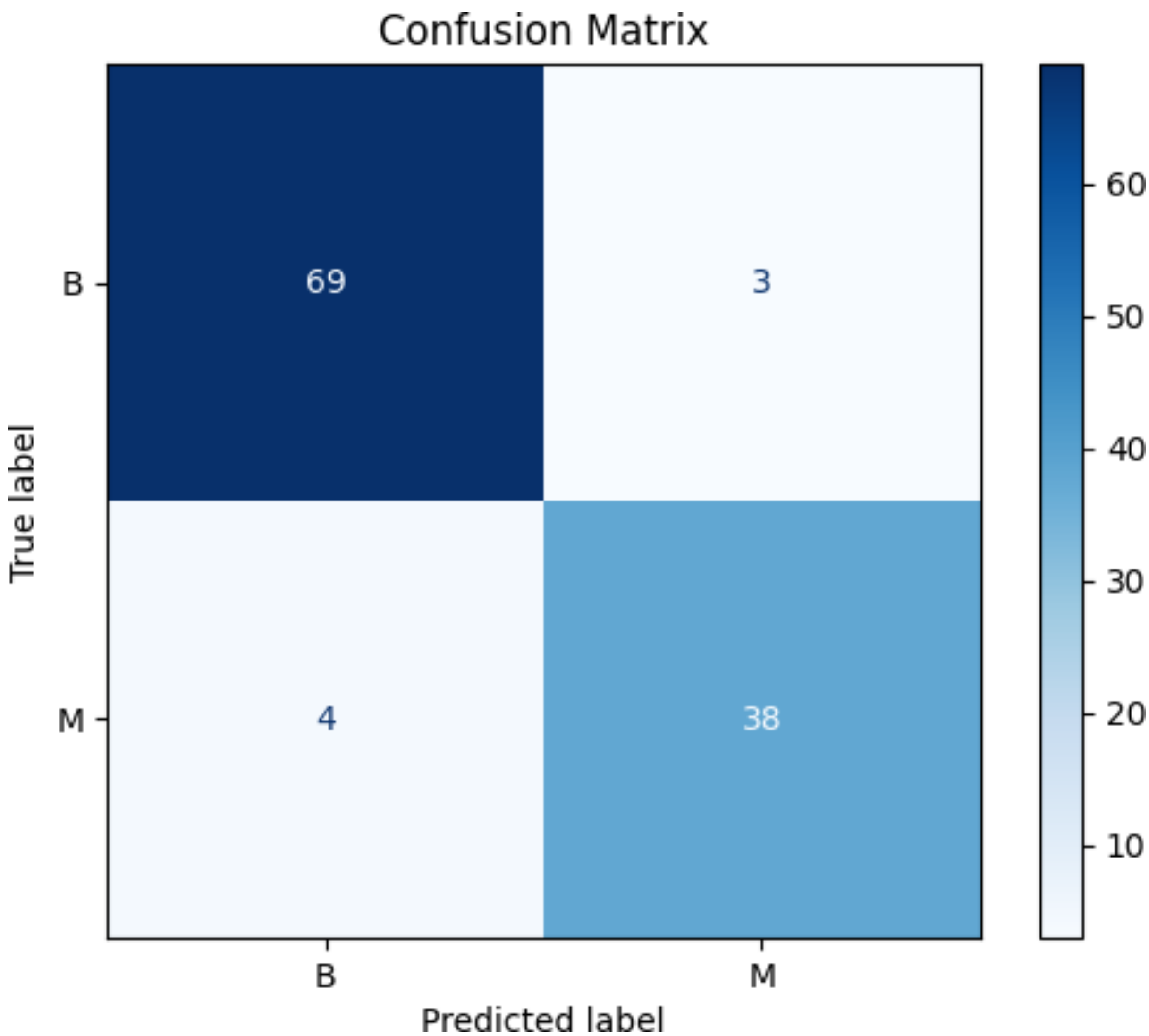

Figure 1: Decision Tree Confusion Matrix

ROC/AUC Curve

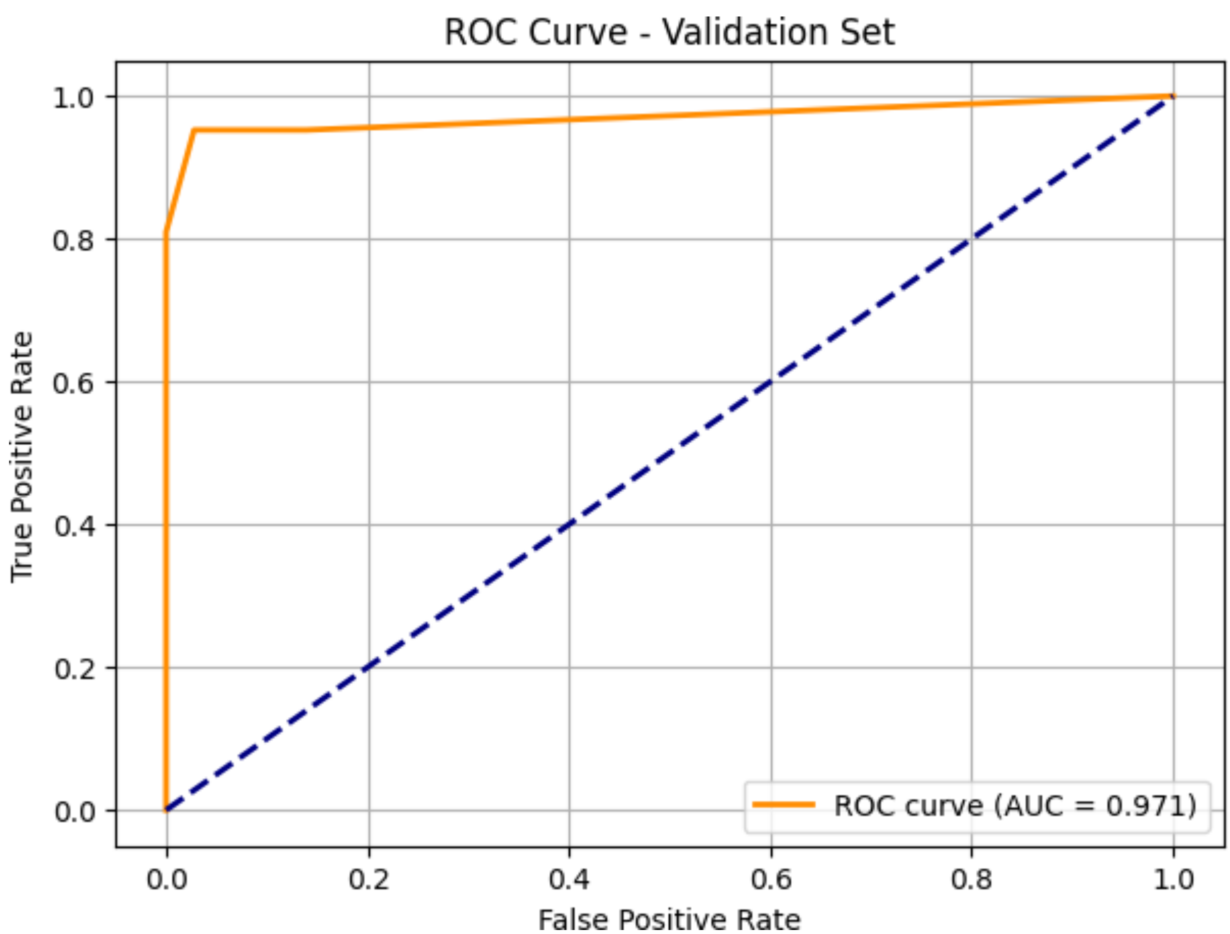

Figure 2: ROC/AUC Curve

Feature Importance Visuals

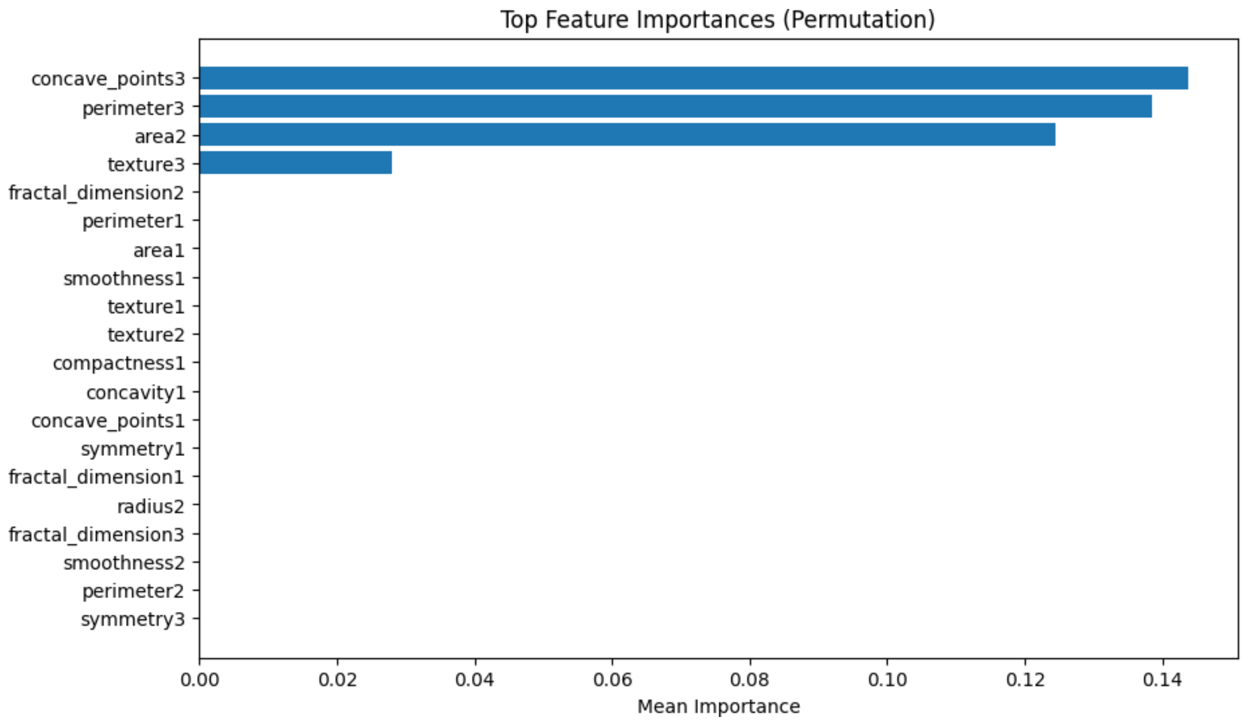

Figure 3: Decision Tree Feature Importance Visuals

AdaBoost

Confusion Matrix

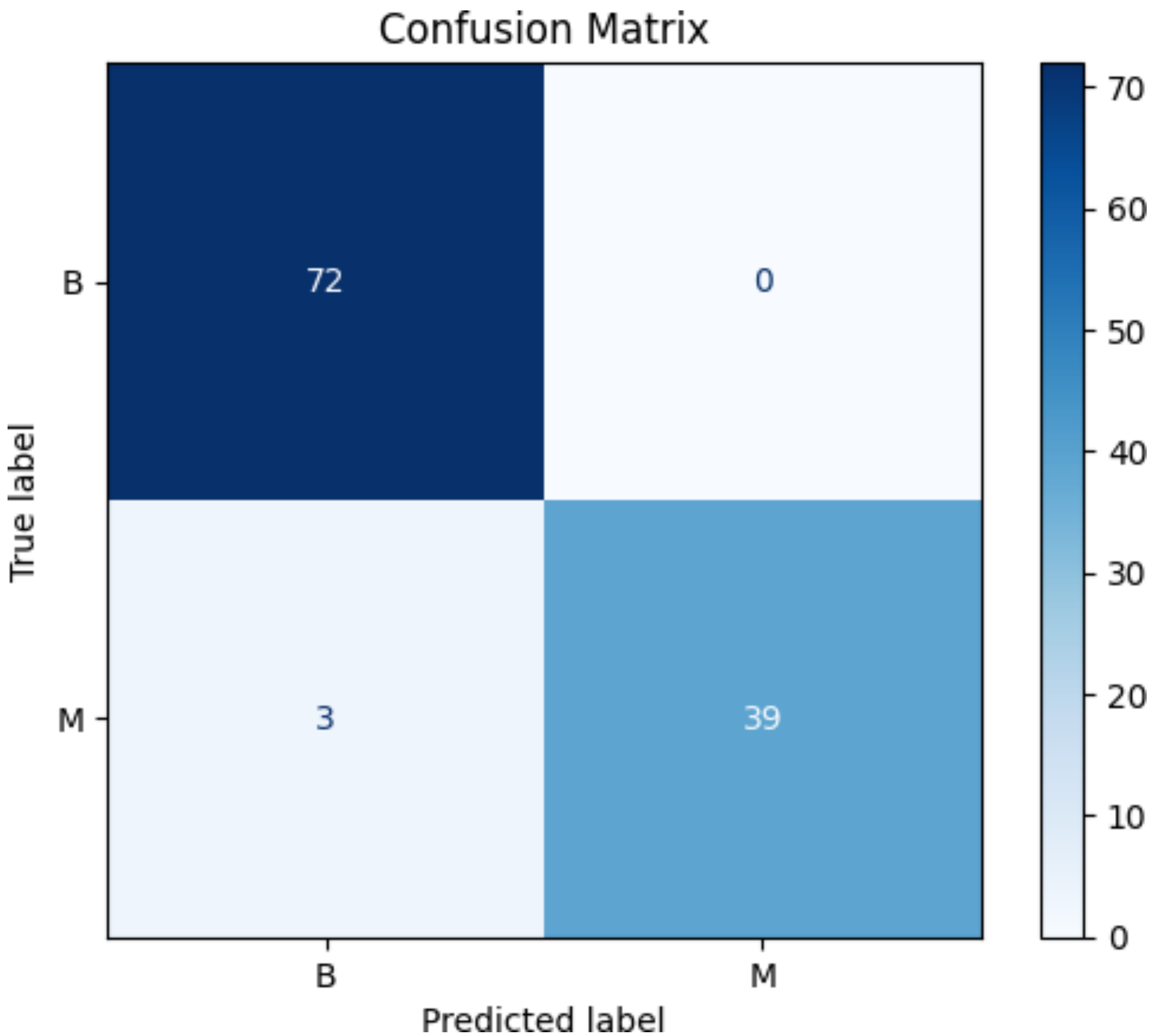

Figure 4: AdaBoost Confusion Matrix

ROC/AUC Curve

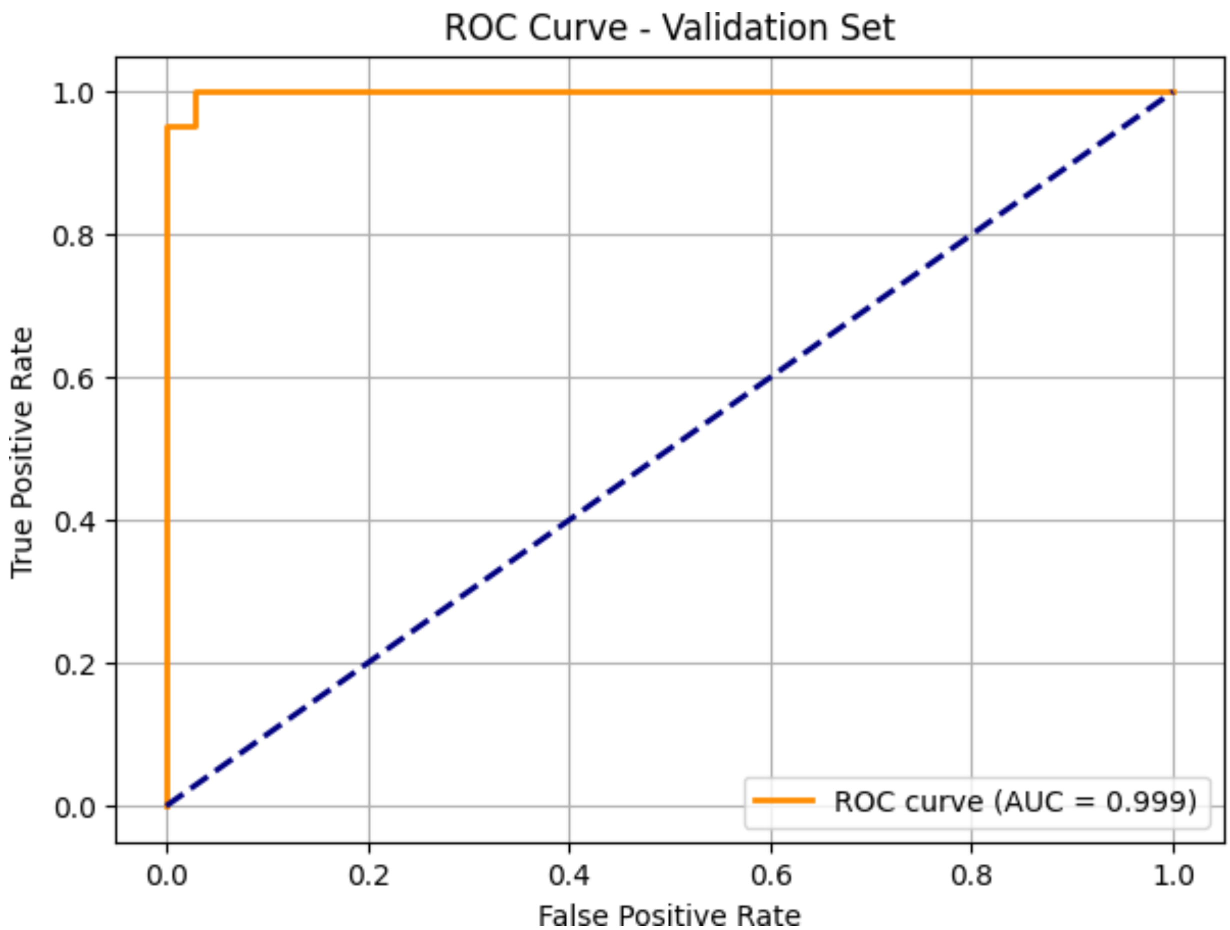

Figure 5: ROC/AUC Curve

Feature Importance Visuals

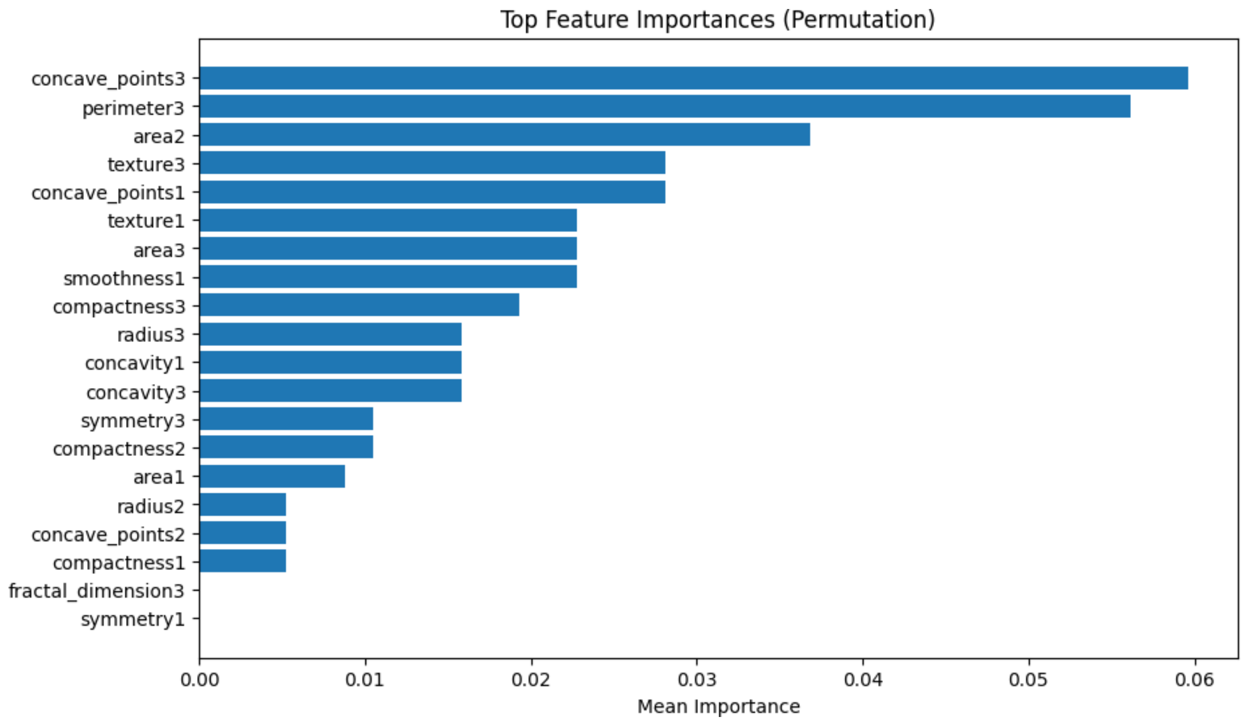

Figure 6: AdaBoost Feature Importance Visuals

Gradient Boosting

Confusion Matrix

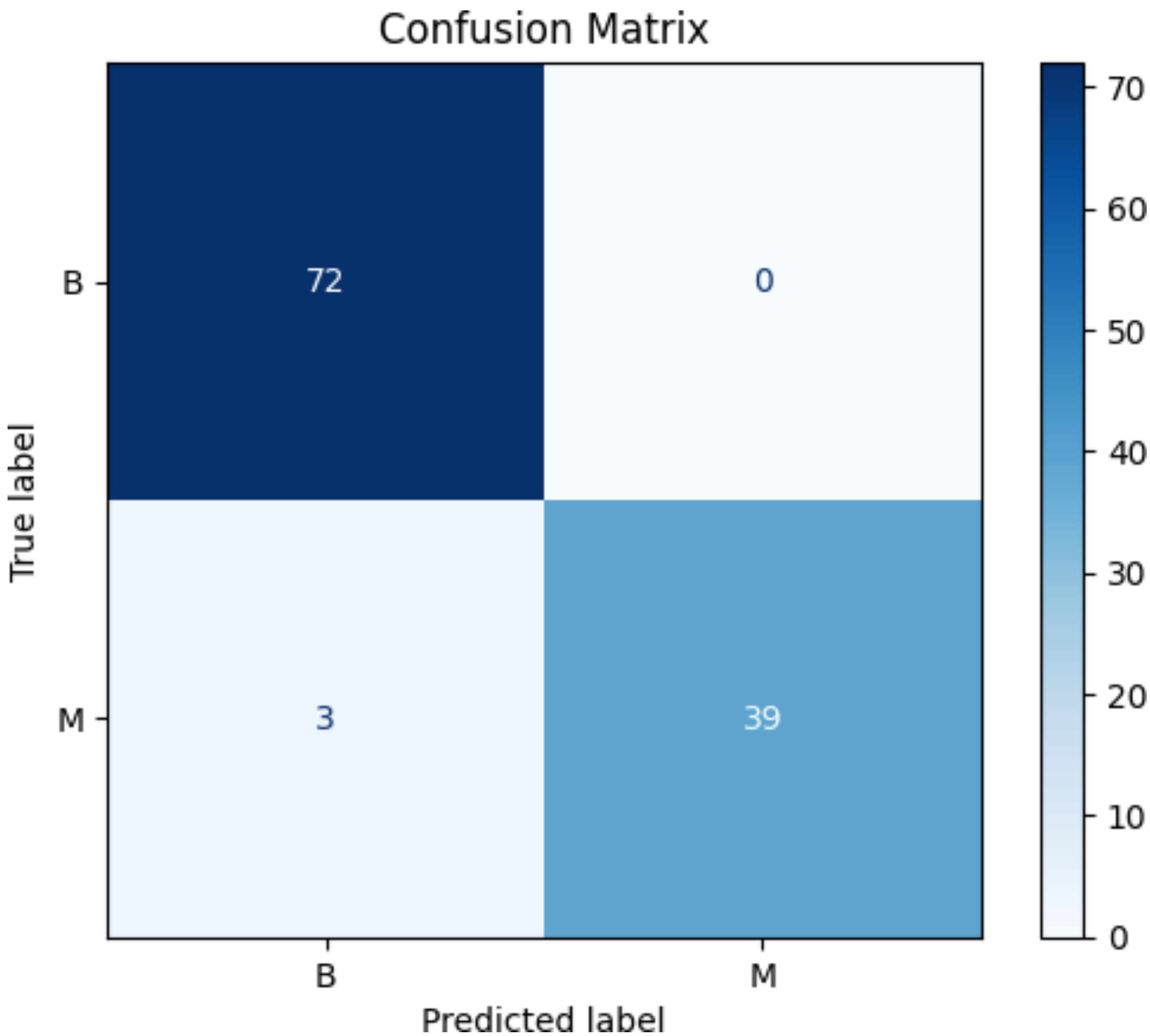

Figure 7: Gradient Boosting Confusion Matrix

ROC/AUC Curve

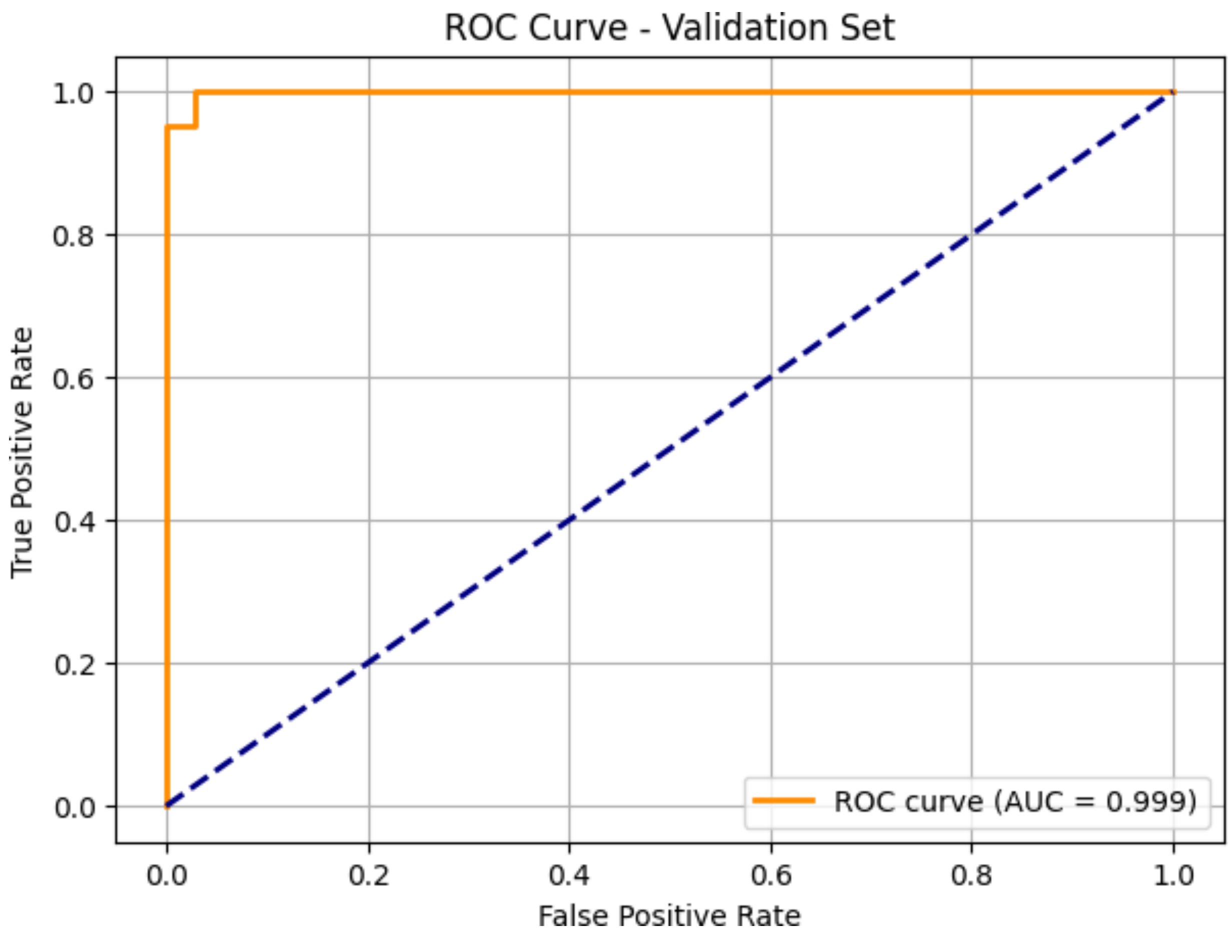

Figure 8: ROC/AUC Curve

Feature Importance Visuals

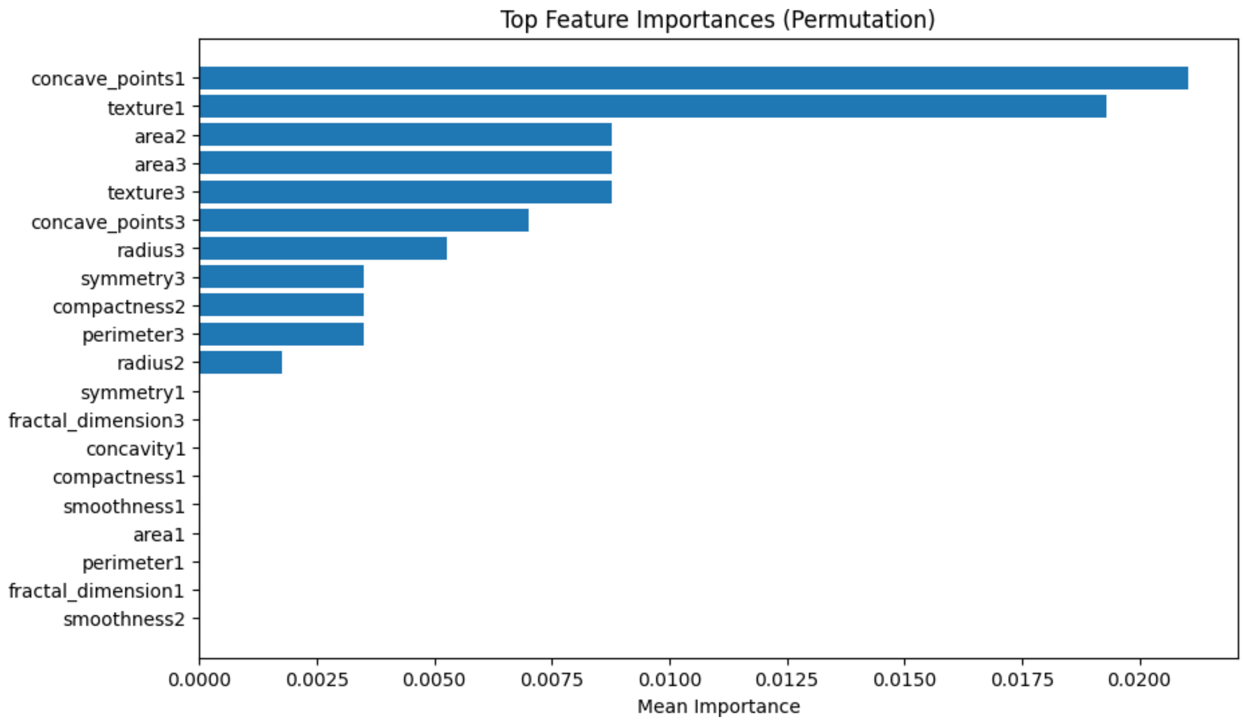

Figure 9: Gradient Boosting Feature Importance Visuals

XGBoost

Confusion Matrix

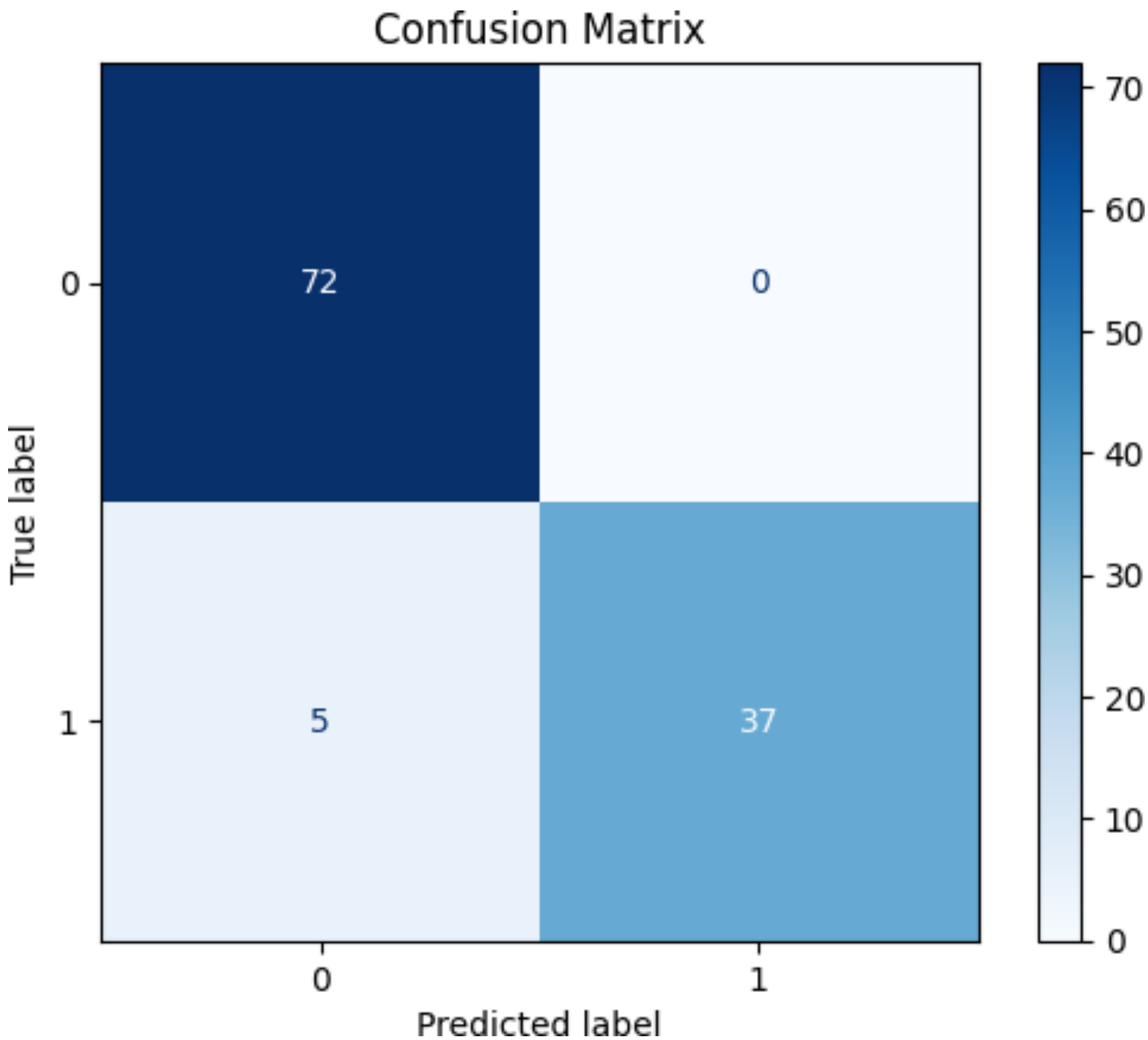

Figure 10: XGBoost Confusion Matrix

ROC/AUC Curve

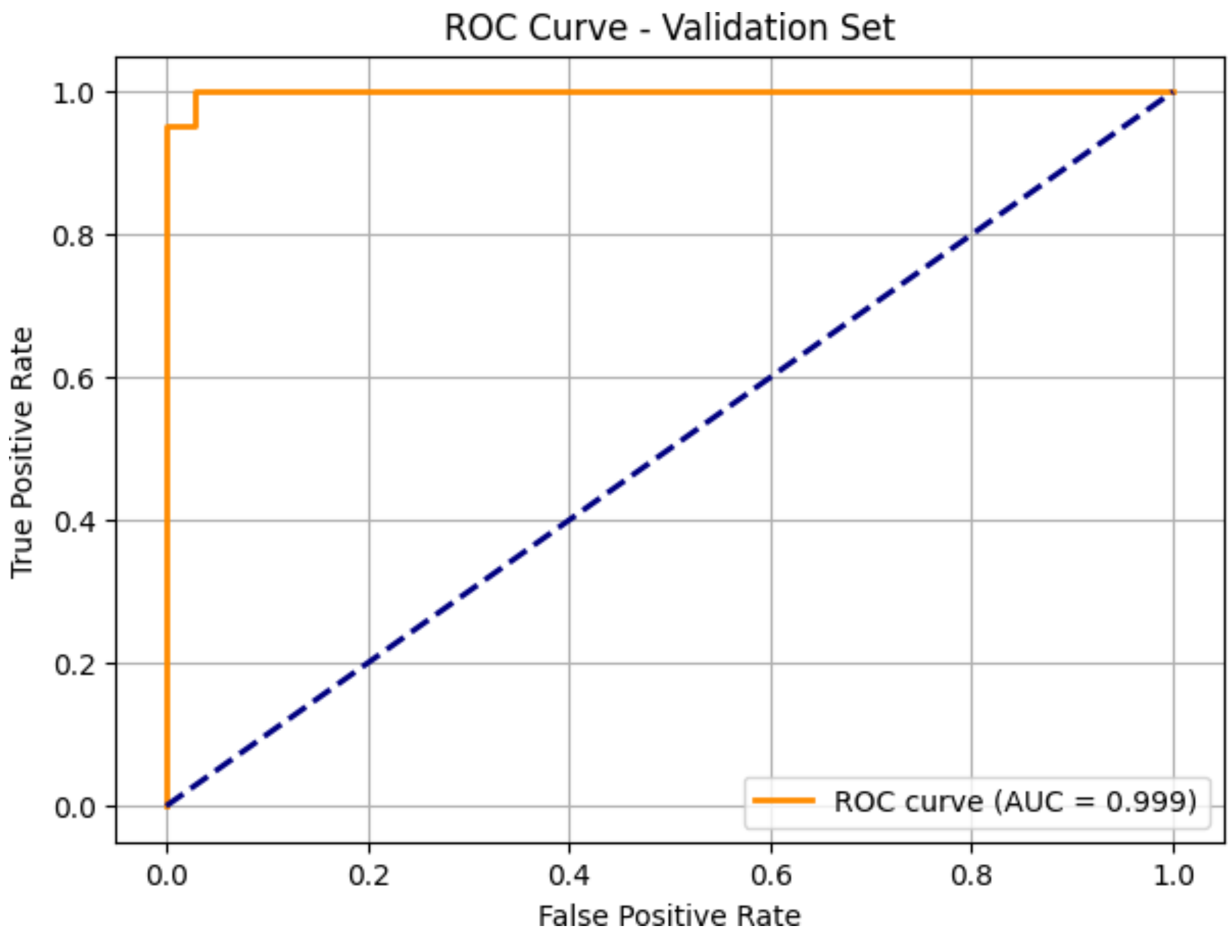

Figure 11: ROC/AUC Curve

Feature Importance Visuals

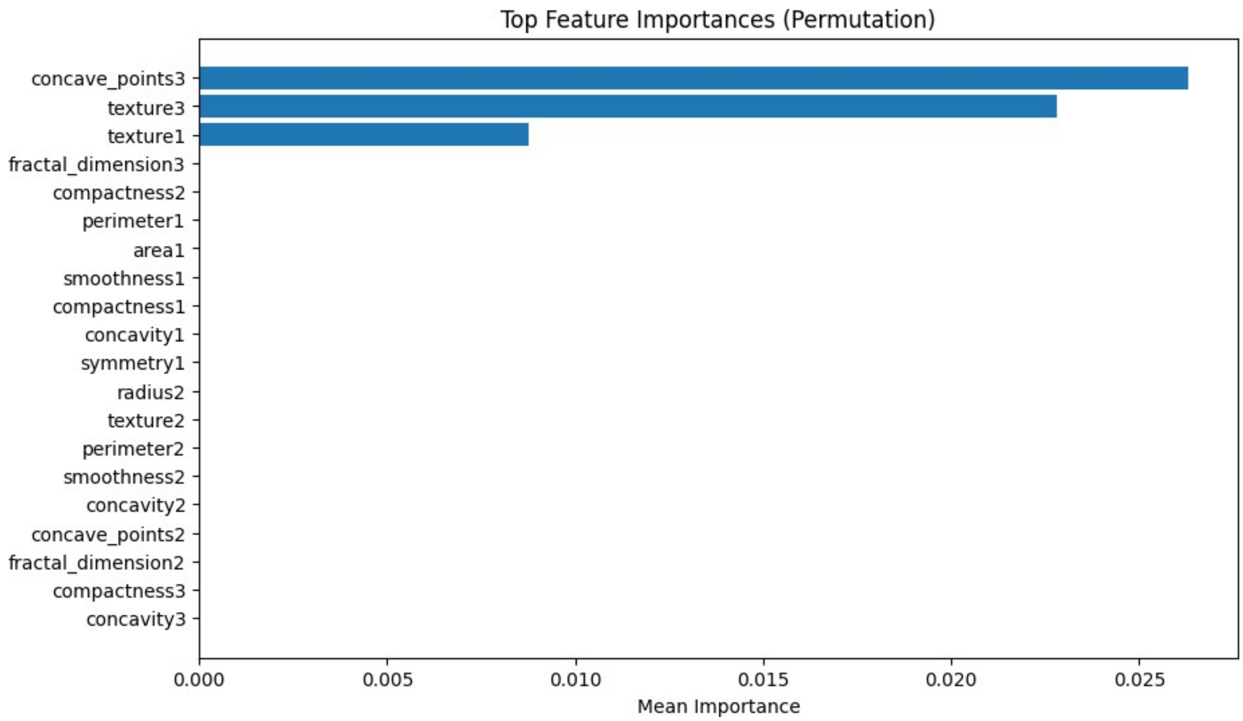

Figure 12: XGBoost Feature Importance Visuals

Random Forest

Confusion Matrix

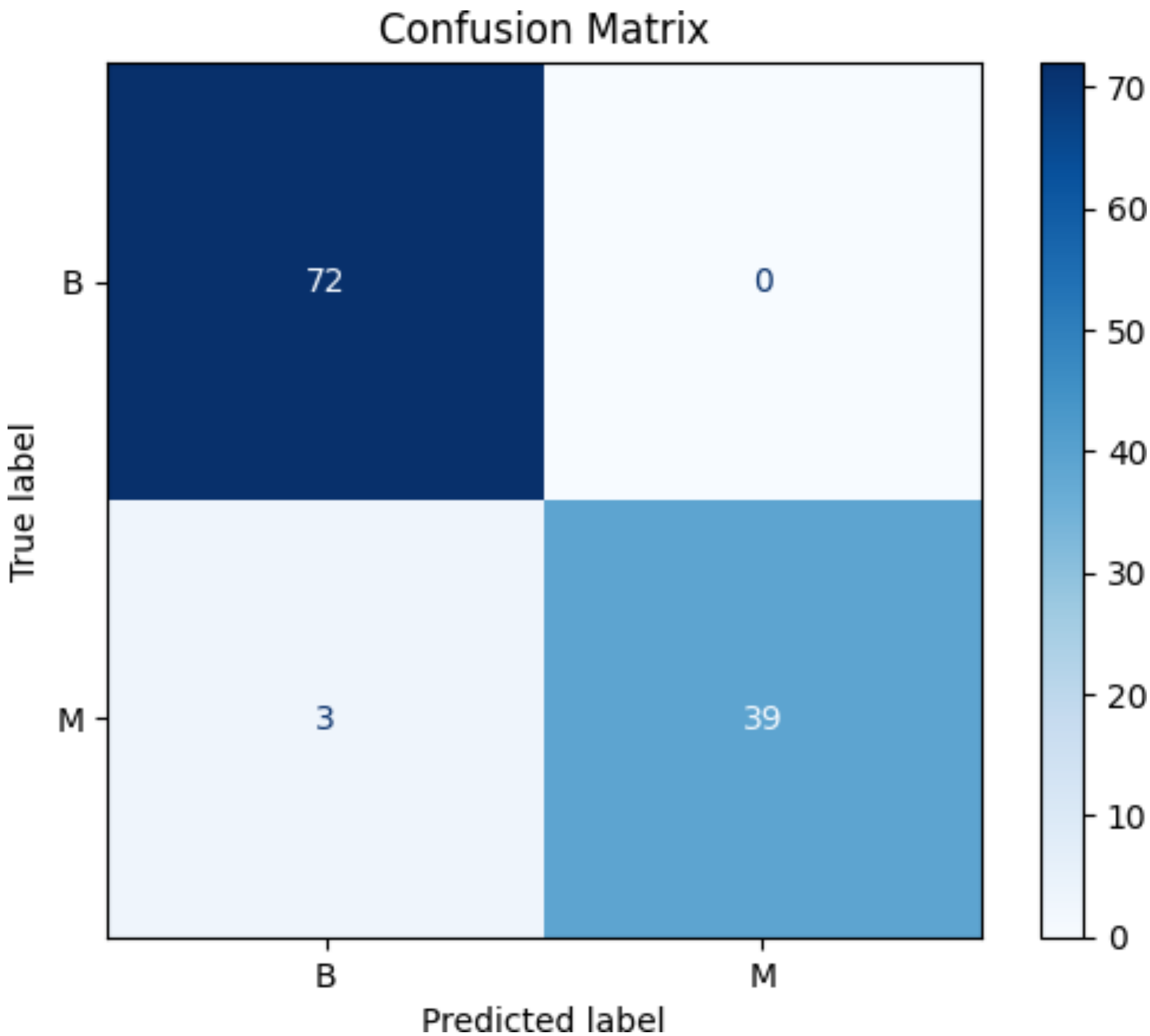

Figure 13: Random Forest Confusion Matrix

ROC/AUC Curve

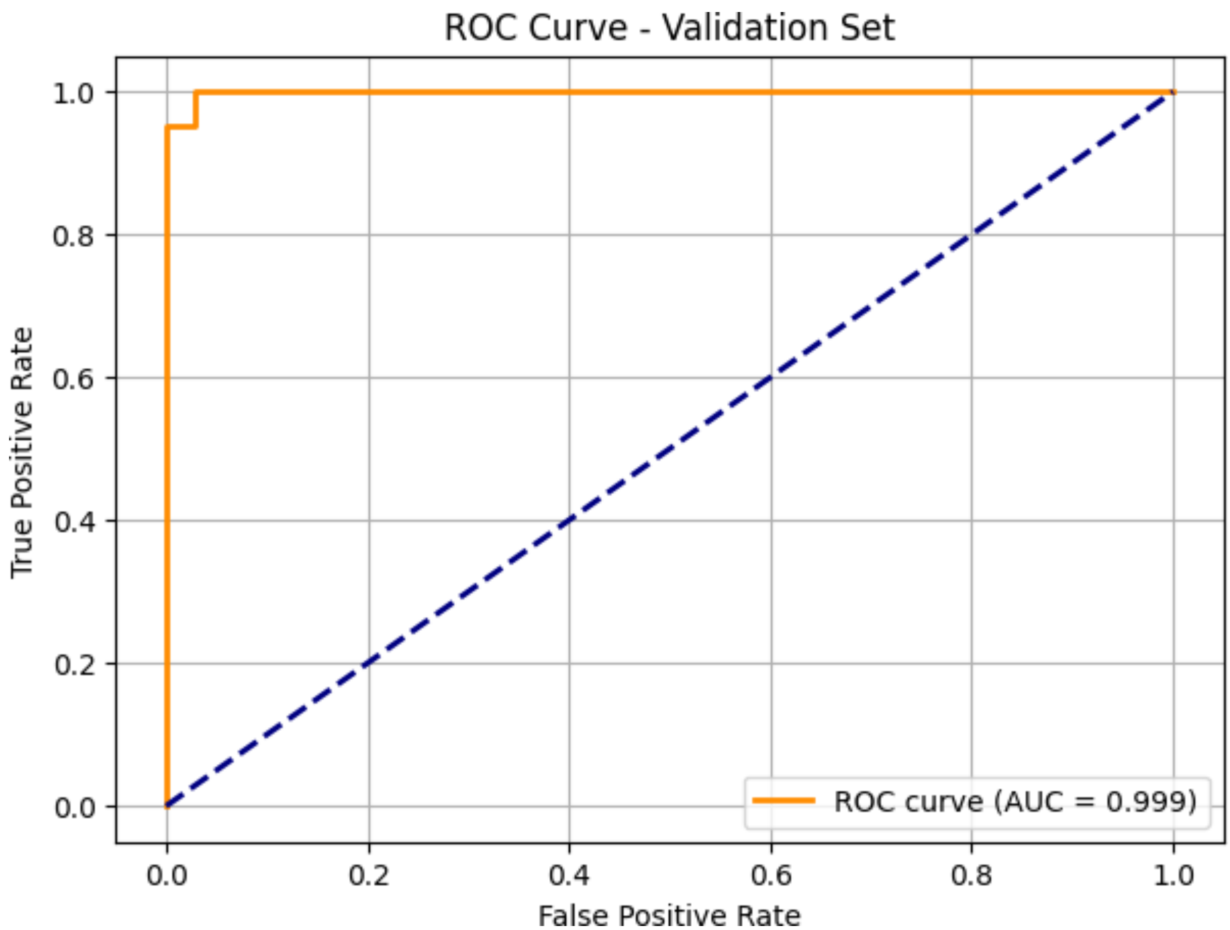

Figure 14: ROC/AUC Curve

Feature Importance Visuals

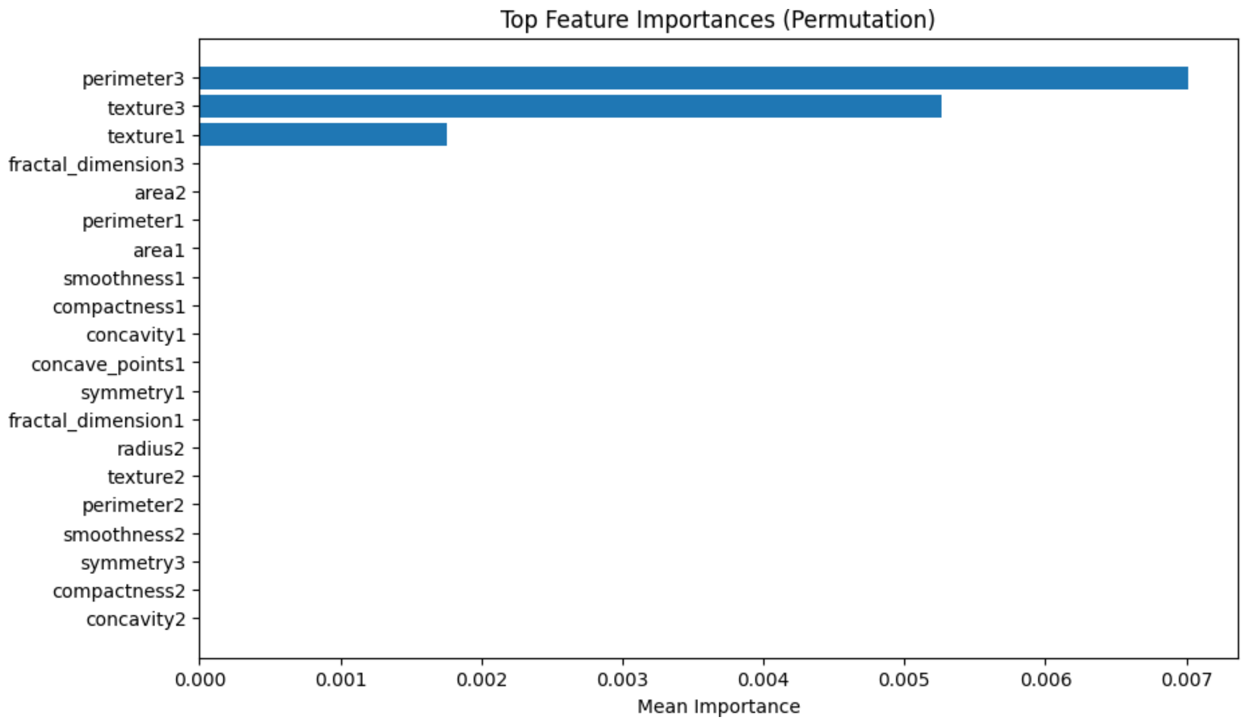

Figure 15: Random Forest Feature Importance Visuals

Stacked Model

Confusion Matrix

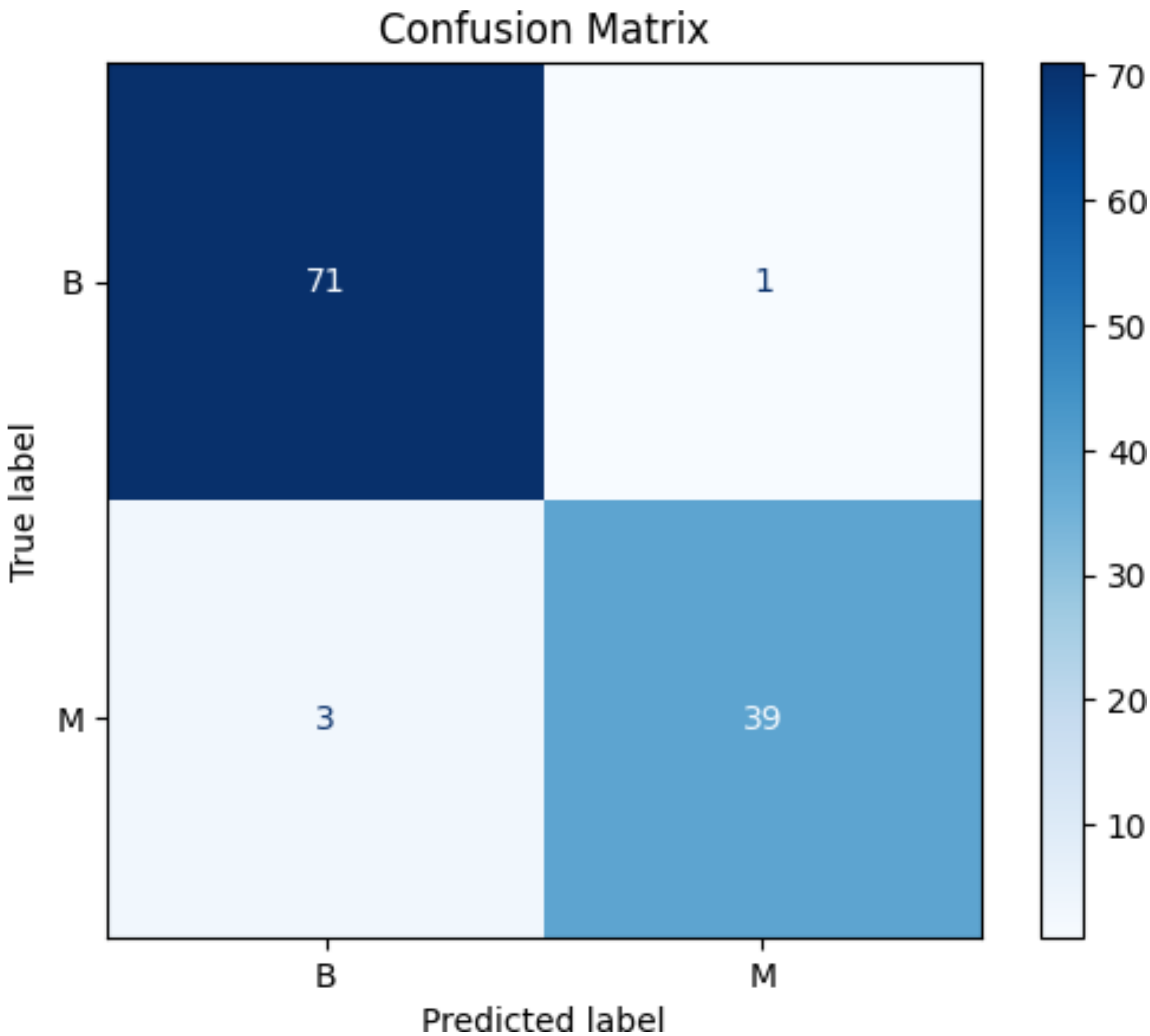

Figure 16: Stacked Model Confusion Matrix

ROC/AUC Curve

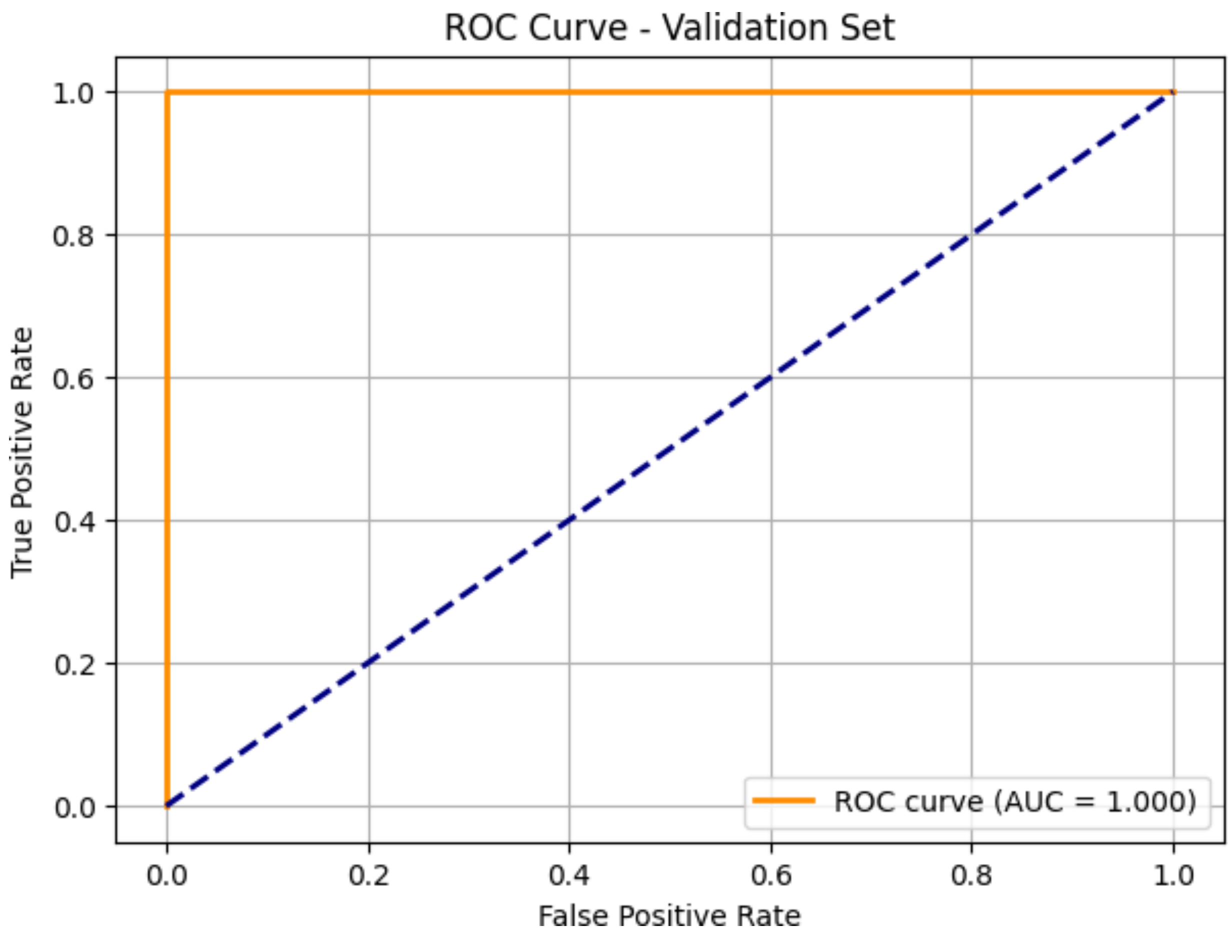

Figure 17: ROC/AUC Curve

Feature Importance Visuals

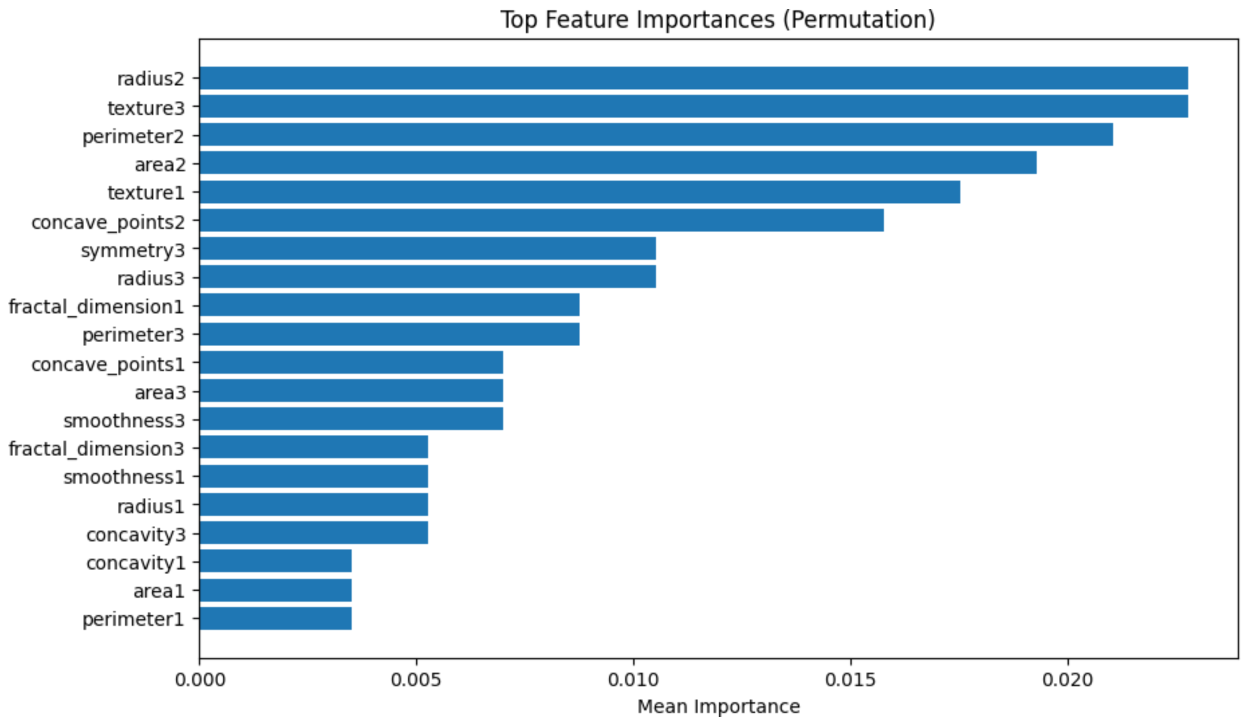

Figure 18: Stacked Model Feature Importance Visuals

Observation Questions and Answers

1. **Which model achieved the best validation accuracy among all six methods?** Stacking achieved the highest validation accuracy, slightly outperforming Gradient Boosting and AdaBoost.

2. **How does Decision Tree performance compare to ensemble methods?** The single Decision Tree performed the worst overall. Ensemble methods such as Random Forest, AdaBoost, and Gradient Boosting significantly improved accuracy and reduced variance.

3. **Did the Random Forest benefit from tuning** max_depth **or** n estimators**?** Yes. Increasing n_estimators stabilized performance and reduced variance, while tuning max_depth prevented overfitting, leading to better validation accuracy.

4. **Which model showed the best generalization? Any signs of overfitting?** Gradient Boosting and Random Forest generalized well. The single Decision Tree showed overfitting (high training accuracy but lower validation accuracy).

5. **Did stacking improve performance over the base models?** Yes. Stacking leveraged the strengths of multiple base learners and achieved the best overall performance, showing an improvement over individual models.

Learning Outcomes:

- Gained practical experience in preprocessing data including handling missing values and outliers.
- Learned to train and evaluate ensemble models such as Decision Tree, AdaBoost, Gradient Boosting, XGBoost, Random Forest, and Stacked Models.
- Understood the role of hyperparameter tuning and 5-Fold Cross-Validation in improving model performance.
- Learned the importance of evaluation metrics such as Accuracy and F1 Score along with visualizations (Confusion Matrix, ROC/AUC Curve, Feature Importances).