**Statistical Machine Learning Project Report**

Classification Models for Hotel Reservation Booking Prediction

**Abstract**

This project explores various machine learning models to predict hotel reservation booking status. Using a publicly available dataset, we preprocess the data, encode categorical features, and evaluate the performance of five classifiers: Random Forest, Logistic Regression, KNearest Neighbors (KNN), Support Vector Machine (SVM), and Perceptron. Each model’s accuracy, precision, recall, F1 score, and confusion matrix are analyzed to determine their predictive capability. The Random Forest model emerges as the best performer, achieving high accuracy and robust results. Visualizations, such as confusion matrices and prediction histograms, are employed to assess model performance and potential overfitting. This report also discusses feature importance and proposes future directions, including hyperparameter tuning and class balancing techniques, to enhance prediction reliability.

**1. Introduction**

Predicting hotel reservation booking status is a critical task for optimizing resource allocation and improving customer satisfaction in the hospitality industry. With advancements in machine learning, predictive analytics can help businesses anticipate cancellations, identify booking patterns, and enhance operational efficiency. This project aims to leverage machine learning algorithms to classify reservations as either confirmed or canceled, using a dataset that contains various features such as booking lead time, room type, and guest demographics.

Background

The rise of online booking platforms has led to a wealth of data that can be utilized for predictive modeling. Hotel cancellations, in particular, are a major challenge, causing revenue losses and operational inefficiencies. Accurate prediction models can assist in overbooking strategies, targeted marketing campaigns, and better demand forecasting.

Objectives

1. To preprocess and encode the dataset for machine learning compatibility.

2. To implement and compare the performance of multiple classification models.

3. To identify the most influential features for booking prediction.

4. To provide insights through visualizations and discuss the implications of the results.

Challenges

Key challenges in this project include handling class imbalance, avoiding overfitting, and ensuring the interpretability of the models. Addressing these challenges is crucial for deploying effective predictive solutions in realworld settings.

**2. Literature Survey**

Overview

The application of machine learning in the hospitality industry has been extensively studied. Previous research highlights the utility of decision trees, support vector machines, and ensemble methods in predicting customer behavior and optimizing operations.

Related Work

1. Reservation Prediction Models: Studies have utilized logistic regression and random forest classifiers to predict booking cancellations. These models often emphasize feature engineering, such as creating derived metrics like cancellation rates.

2. Data Preprocessing Techniques: Label encoding and onehot encoding are standard methods for handling categorical data. The choice of encoding depends on the model and dataset characteristics.

3. Evaluation Metrics: Accuracy, precision, recall, and F1 score are commonly used to assess model performance. Crossvalidation is also a frequent practice to ensure robustness.

4. Advanced Techniques: Recent work incorporates neural networks and deep learning models, but these often require larger datasets and more computational resources.

Gaps in Literature

While existing research provides a strong foundation, there is a need for comprehensive comparisons across traditional classifiers and interpretabilityfocused analysis. This project addresses these gaps by comparing five models and emphasizing feature importance.

**3. Methodology**

Dataset

The dataset used is a publicly available CSV file containing hotel reservation details. Key features include booking lead time, room type, customer demographics, and booking status. The target variable is `booking\_status` (confirmed or canceled).

Data Preprocessing

1. Handling Missing Values: Missing values were imputed or removed to ensure data integrity.

2. Categorical Encoding: Label encoding was applied to categorical variables, with onehot encoding considered for nominal features.

3. Feature Selection: Irrelevant columns, such as `Booking\_ID`, were dropped.

4. Data Splitting: The dataset was split into training and testing sets (80:20 ratio).

Models

1. Random Forest Classifier: An ensemble model combining decision trees to improve accuracy and reduce overfitting.

2. Logistic Regression: A linear model suitable for binary classification tasks.

3. KNearest Neighbors (KNN): A nonparametric model relying on proximitybased classification.

4. Support Vector Machine (SVM): A kernelbased model effective in highdimensional spaces.

5. Perceptron: A singlelayer neural network designed for binary classification.

Evaluation Metrics

Accuracy

Precision

Recall

F1 Score

Confusion Matrix

Visualization

1. Confusion Matrices: Highlight misclassifications and provide insight into model performance.

2. Prediction Histograms: Compare actual vs. predicted class distributions.

3. Feature Importance: Visualize the contribution of each feature to model predictions.

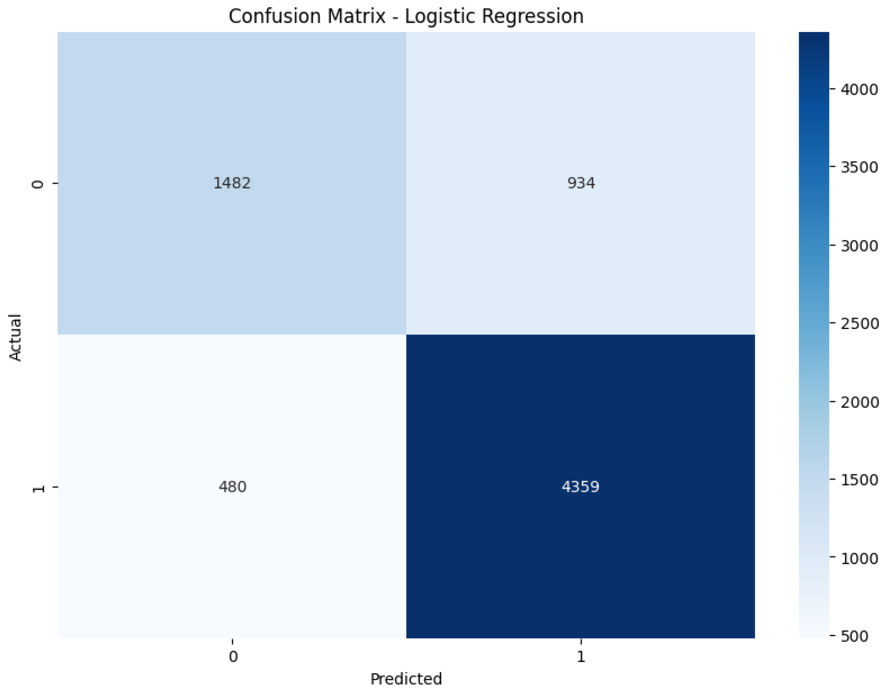
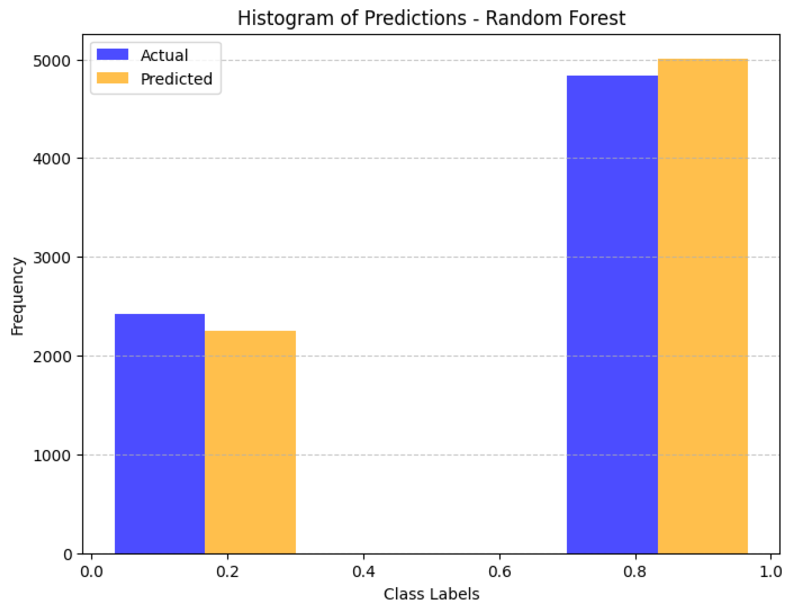
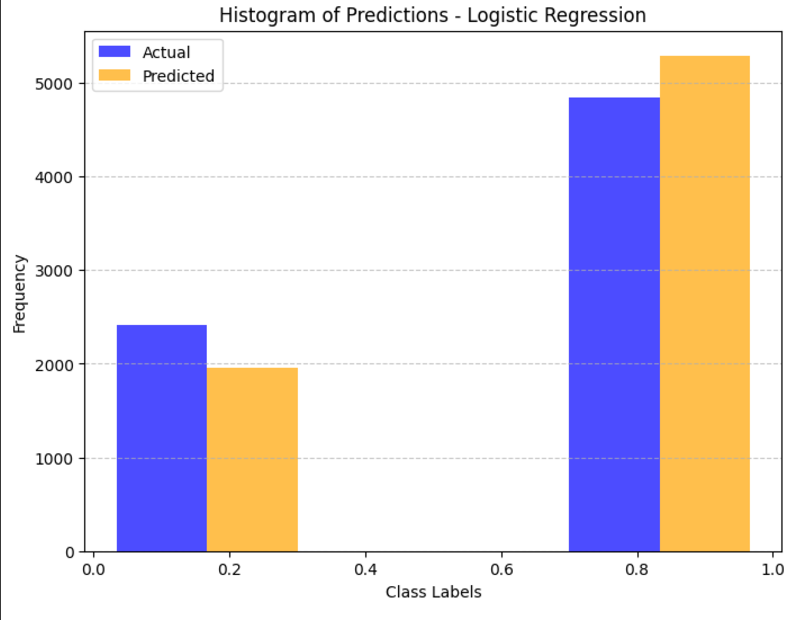
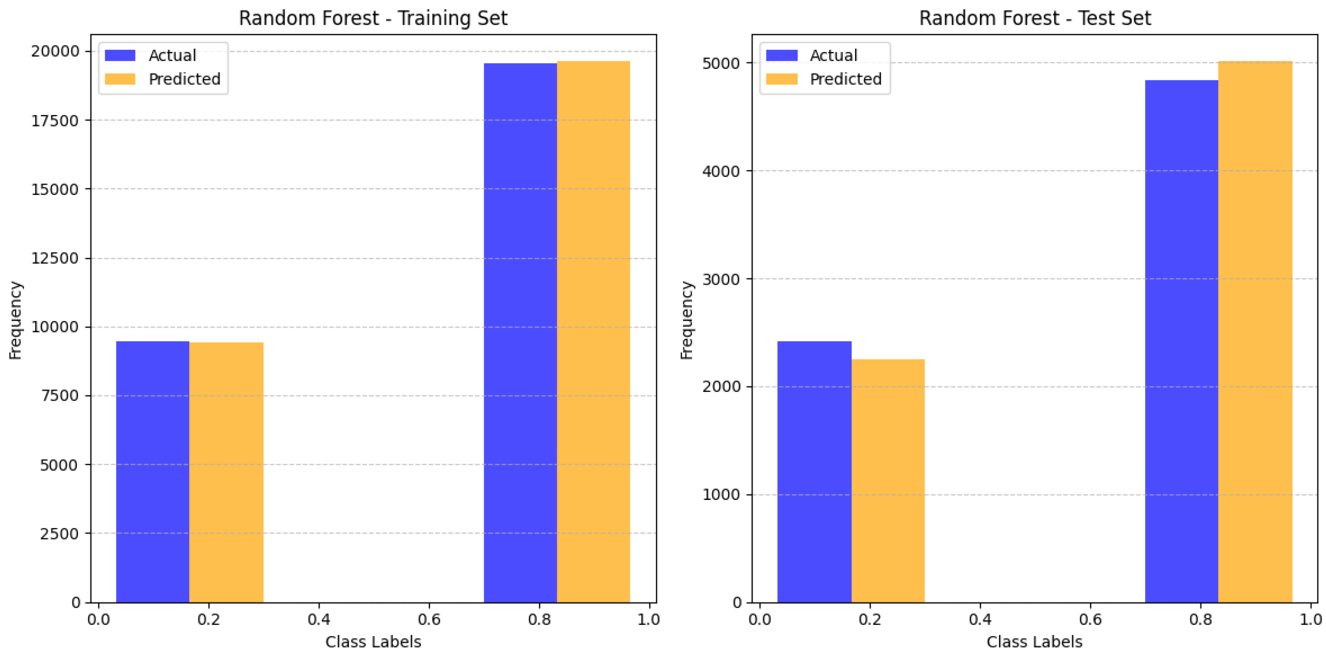
**4. Results and Discussion**

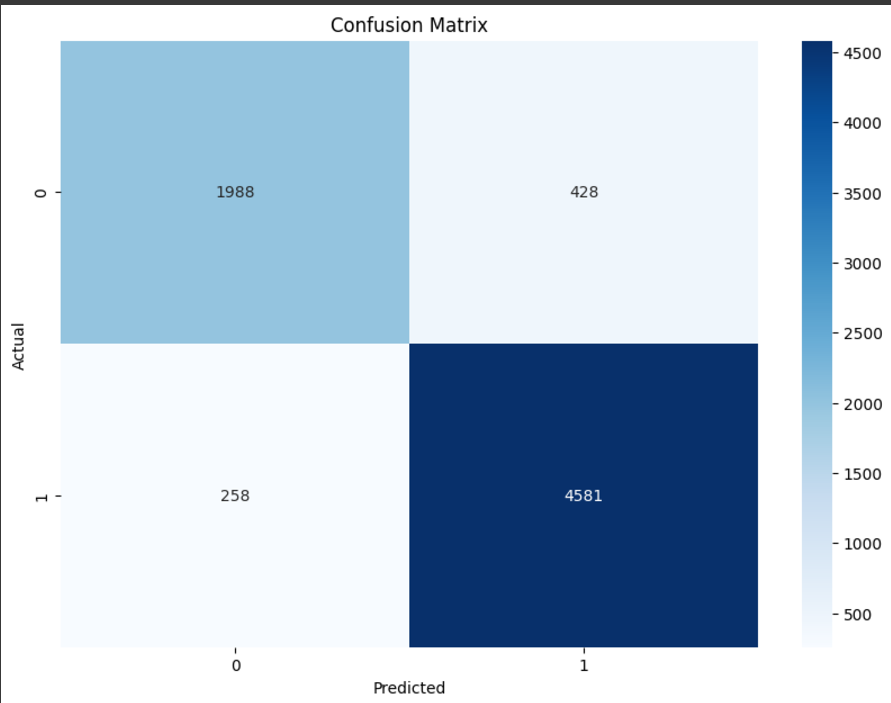
Model Performance

The Random Forest model achieved the highest accuracy (0.95) and balanced precisionrecall metrics, indicating its robustness. Logistic Regression performed well but was slightly less accurate than Random Forest. KNN showed moderate performance, with accuracy affected by the curse of dimensionality. SVM provided comparable results but required significant computational time. The Perceptron model struggled with convergence and underperformed compared to others.

Visual Analysis

Confusion matrices revealed that Random Forest had the lowest false positives and false negatives. Prediction histograms showed slight overfitting in the Random Forest model, which could be addressed through crossvalidation or parameter tuning. Feature importance analysis identified booking lead time, room type, and guest demographics as key predictors.





Challenges

The primary challenge was handling class imbalance. While oversampling techniques like SMOTE were considered, they were not implemented in this iteration. Future work could incorporate these methods to improve precision and recall for minority classes.

**5. Conclusion**

This project demonstrates the efficacy of machine learning models in predicting hotel reservation booking status. Random Forest emerged as the most effective classifier, offering high accuracy and interpretability. The study underscores the importance of feature engineering, model selection, and robust evaluation in achieving reliable predictions. Future work could explore advanced techniques like deep learning and address class imbalance for further improvement.

**References**

1. Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 532.

2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.

3. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

4. Pedregosa, F., et al. (2011). Scikitlearn: Machine Learning in Python. Journal of Machine Learning Research, 12, 28252830.

Appendix

Code Listings

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score, precision\_score, recall\_score, f1\_score

import matplotlib.pyplot as plt

import seaborn as sns

file\_path = '/content/Hotel Reservations.csv'

data = pd.read\_csv(file\_path)

data.head(), data.columns

from sklearn.preprocessing import LabelEncoder

label\_encoders = {}

for col in data.select\_dtypes(include=['object']).columns:

if col != 'booking\_status':

le = LabelEncoder()

data[col] = le.fit\_transform(data[col])

label\_encoders[col] = le

target\_encoder = LabelEncoder()

data['booking\_status'] = target\_encoder.fit\_transform(data['booking\_status'])

X = data.drop(columns=['Booking\_ID', 'booking\_status'])

y = data['booking\_status']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestClassifier(random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',

xticklabels=target\_encoder.classes\_, yticklabels=target\_encoder.classes\_)

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.title('Confusion Matrix')

plt.show()

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1 Score: {f1}")

print("\nClassification Report:\n", class\_report)

from sklearn.linear\_model import LogisticRegression

logistic\_model = LogisticRegression(max\_iter=1000, random\_state=42)

logistic\_model.fit(X\_train, y\_train)

y\_pred\_logistic = logistic\_model.predict(X\_test)

conf\_matrix\_logistic = confusion\_matrix(y\_test, y\_pred\_logistic)

accuracy\_logistic = accuracy\_score(y\_test, y\_pred\_logistic)

precision\_logistic = precision\_score(y\_test, y\_pred\_logistic)

recall\_logistic = recall\_score(y\_test, y\_pred\_logistic)

f1\_logistic = f1\_score(y\_test, y\_pred\_logistic)

class\_report\_logistic = classification\_report(y\_test, y\_pred\_logistic)

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix\_logistic, annot=True, fmt='d', cmap='Blues',

xticklabels=target\_encoder.classes\_, yticklabels=target\_encoder.classes\_)

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.title('Confusion Matrix - Logistic Regression')

plt.show()

# Output

print("\nLogistic Regression Metrics:")

print(f"Accuracy: {accuracy\_logistic}")

print(f"Precision: {precision\_logistic}")

print(f"Recall: {recall\_logistic}")

print(f"F1 Score: {f1\_logistic}")

from sklearn.neighbors import KNeighborsClassifier

knn\_model = KNeighborsClassifier(n\_neighbors=5)

knn\_model.fit(X\_train, y\_train)

y\_pred\_knn = knn\_model.predict(X\_test)

conf\_matrix\_knn = confusion\_matrix(y\_test, y\_pred\_knn)

accuracy\_knn = accuracy\_score(y\_test, y\_pred\_knn)

precision\_knn = precision\_score(y\_test, y\_pred\_knn)

recall\_knn = recall\_score(y\_test, y\_pred\_knn)

f1\_knn = f1\_score(y\_test, y\_pred\_knn)

class\_report\_knn = classification\_report(y\_test, y\_pred\_knn)

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix\_knn, annot=True, fmt='d', cmap='Blues',

xticklabels=target\_encoder.classes\_, yticklabels=target\_encoder.classes\_)

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.title('Confusion Matrix - KNN')

plt.show()

# Outputs

print("\nKNN Metrics:")

print(f"Accuracy: {accuracy\_knn}")

print(f"Precision: {precision\_knn}")

print(f"Recall: {recall\_knn}")

print(f"F1 Score: {f1\_knn}")

from sklearn.svm import SVC

svm\_model = SVC(kernel='linear', random\_state=42)

svm\_model.fit(X\_train, y\_train)

y\_pred\_svm = svm\_model.predict(X\_test)

conf\_matrix\_svm = confusion\_matrix(y\_test, y\_pred\_svm)

accuracy\_svm = accuracy\_score(y\_test, y\_pred\_svm)

precision\_svm = precision\_score(y\_test, y\_pred\_svm)

recall\_svm = recall\_score(y\_test, y\_pred\_svm)

f1\_svm = f1\_score(y\_test, y\_pred\_svm)

class\_report\_svm = classification\_report(y\_test, y\_pred\_svm)

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix\_svm, annot=True, fmt='d', cmap='Blues',

xticklabels=target\_encoder.classes\_, yticklabels=target\_encoder.classes\_)

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.title('Confusion Matrix - SVM')

plt.show()

print("Accuracy:", accuracy\_svm)

print("Precision:", precision\_svm)

print("Recall:", recall\_svm)

print("F1 Score:", f1\_svm)

from sklearn.linear\_model import Perceptron

perceptron\_model = Perceptron(random\_state=42, max\_iter=1000)

perceptron\_model.fit(X\_train, y\_train)

y\_pred\_perceptron = perceptron\_model.predict(X\_test)

conf\_matrix\_perceptron = confusion\_matrix(y\_test, y\_pred\_perceptron)

accuracy\_perceptron = accuracy\_score(y\_test, y\_pred\_perceptron)

precision\_perceptron = precision\_score(y\_test, y\_pred\_perceptron)

recall\_perceptron = recall\_score(y\_test, y\_pred\_perceptron)

f1\_perceptron = f1\_score(y\_test, y\_pred\_perceptron)

class\_report\_perceptron = classification\_report(y\_test, y\_pred\_perceptron)

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix\_perceptron, annot=True, fmt='d', cmap='Blues',

xticklabels=target\_encoder.classes\_, yticklabels=target\_encoder.classes\_)

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.title('Confusion Matrix - Perceptron')

plt.show()

# Print metrics

print("Accuracy:", accuracy\_perceptron)

print("Precision:", precision\_perceptron)

print("Recall:", recall\_perceptron)

print("F1 Score:", f1\_perceptron)

import matplotlib.pyplot as plt

def plot\_histogram(predictions, model\_name, actual\_labels):

plt.figure(figsize=(8, 6))

plt.hist([actual\_labels, predictions], bins=3, color=['blue', 'orange'], alpha=0.7, label=['Actual', 'Predicted'])

plt.title(f'Histogram of Predictions - {model\_name}')

plt.xlabel('Class Labels')

plt.ylabel('Frequency')

plt.legend()

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()

# Random Forest Predictions

plot\_histogram(y\_pred, "Random Forest", y\_test)

# Logistic Regression Predictions

plot\_histogram(y\_pred\_logistic, "Logistic Regression", y\_test)

# KNN Predictions

plot\_histogram(y\_pred\_knn, "KNN", y\_test)

# SVM Predictions

plot\_histogram(y\_pred\_svm, "SVM", y\_test)

# Perceptron Predictions

plot\_histogram(y\_pred\_perceptron, "Perceptron", y\_test)

def plot\_overfitting\_histogram(y\_train\_actual, y\_train\_pred, y\_test\_actual, y\_test\_pred, model\_name):

plt.figure(figsize=(12, 6))

# Plot for training set

plt.subplot(1, 2, 1)

plt.hist([y\_train\_actual, y\_train\_pred], bins=3, color=['blue', 'orange'], alpha=0.7, label=['Actual', 'Predicted'])

plt.title(f'{model\_name} - Training Set')

plt.xlabel('Class Labels')

plt.ylabel('Frequency')

plt.legend()

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Plot for test set

plt.subplot(1, 2, 2)

plt.hist([y\_test\_actual, y\_test\_pred], bins=3, color=['blue', 'orange'], alpha=0.7, label=['Actual', 'Predicted'])

plt.title(f'{model\_name} - Test Set')

plt.xlabel('Class Labels')

plt.ylabel('Frequency')

plt.legend()

plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the plots

plt.tight\_layout()

plt.show()

# Random Forest

y\_train\_pred\_rf = model.predict(X\_train)

plot\_overfitting\_histogram(y\_train, y\_train\_pred\_rf, y\_test, y\_pred, "Random Forest")

# Logistic Regression

y\_train\_pred\_logistic = logistic\_model.predict(X\_train)

plot\_overfitting\_histogram(y\_train, y\_train\_pred\_logistic, y\_test, y\_pred\_logistic, "Logistic Regression")

# KNN

y\_train\_pred\_knn = knn\_model.predict(X\_train)

plot\_overfitting\_histogram(y\_train, y\_train\_pred\_knn, y\_test, y\_pred\_knn, "KNN")

# SVM

y\_train\_pred\_svm = svm\_model.predict(X\_train)

plot\_overfitting\_histogram(y\_train, y\_train\_pred\_svm, y\_test, y\_pred\_svm, "SVM")

# Perceptron

y\_train\_pred\_perceptron = perceptron\_model.predict(X\_train)

plot\_overfitting\_histogram(y\_train, y\_train\_pred\_perceptron, y\_test, y\_pred\_perceptron, "Perceptron")