RR_Project_Report

June 2, 2024

1 REPRODUCIBLE RESEARCH PROJECT

June, 2024

Team Members * Dominik Koterwa * Hüseyin Polat * Nurdan Beşli 457945 * Maciej Lorens 419763

1.1 INTRODUCTION

Our project is centered on replicating and expanding upon the groundbreaking machine learning research detailed in the Asian Journal of Computer and Information Systems (AJCIS), specifically the article titled '**Utilisation of Machine Learning Techniques in Testing and Training of Different Medical Datasets**'. This initiative seeks to deepen our understanding of how machine learning can revolutionize the analysis of medical data, thereby enhancing disease detection and improving diagnostic accuracy across various health conditions.

By engaging with this research, we aim to assess the reproducibility of the results and explore potential enhancements in the methodology. Our study will utilize a similar approach to analyze medical datasets, focusing on maximizing the potential of machine learning in healthcare. Through rigorous testing and training, our objective is to contribute meaningful advancements to the medical field, offering insights that could be vital in developing more effective diagnostic tools and treatments.

1.2 DATASETS

The study we replicated utilized machine learning algorithms to analyze six different healthcare datasets, all sourced from the **UC Irvine Machine Learning Repository**. Each of these datasets varies in the number of attributes and instances, as detailed below.

1.3 METHODOLOGY

1.3.1 DATA PREPROCESSING

For each dataset involved in our study, detailed data preprocessing steps are applied to ensure compatibility with machine learning models and to maintain the integrity of the data analysis. Below are the dataset-specific preprocessing methods implemented:

COVID-19 Dataset * **Binary Remapping:** Each categorical variable representing binary states (e.g., positive '+' or negative '-') is replaced with numerical values where '+' is mapped to 1 and '-' is mapped to 0. * **Numerical Conversion:** All columns are converted to numeric types to facilitate mathematical operations and model fitting. * **Column Renaming:** To clearly indicate the binary nature of the data, columns are renamed by adding a 'is_' prefix, enhancing clarity and manageability in the dataset.

Hepatitis Dataset * **Binary Remapping:** Specific columns that contain categorical data related to the patient's conditions (e.g., 'Steroid', 'Antivirals', 'Fatigue') are converted to binary format, where applicable values are adjusted to 0 and 1 to reflect absence and presence of the condition. * **Mean and Median Imputation:** Missing values are addressed by calculating and substituting the mean and median for each column, allowing for robustness against outliers and skewed data. * **Dropping Missing Data:** Rows containing missing data are removed to avoid the introduction of bias or inaccuracies in the model's predictions. Corresponding entries in the target variable are also excluded to maintain dataset alignment.

Chronic Kidney Disease Dataset * **Binary Mapping:** Categorical descriptions (e.g., 'normal', 'abnormal'; 'present', 'notpresent') in symptoms and diagnostic results are mapped to binary (1, 0) to simplify the input for algorithm processing. * **Mean and Median Imputation:** Implements both mean and median imputation for handling missing values, providing two sets of data for comparative analysis and model robustness. * **Dropping Missing Data:** Ensures that only complete data entries are processed, by excluding rows with any missing values from the analysis, thus maintaining the reliability of the dataset.

Breast Cancer Dataset * **Binary Mapping:** The target variable 'Diagnosis' is converted from its original categorical labels ('M' for malignant, 'B' for benign) into a binary format (1 for malignant, 0 for benign), aligning it with binary classification tasks in machine learning. * **Data Checks and Cleaning:** Verifies data types, checks for outliers, and ensures that all entries are consistent and appropriate for analysis.

Immunotherapy Dataset * **Binary Mapping and Renaming:** Converts 'sex' from a categorical to a binary format and updates the column name to 'is_female' to reflect this binary distinction. * **One-Hot Encoding:** Applies one-hot encoding to the 'Type' column, which involves transforming it into multiple binary columns, each representing a different type of immunotherapy, thereby avoiding ordinal implications in the categorical data. * **Concatenation:** The one-hot encoded columns are then concatenated back to the main dataset, preserving the original data structure while adding new binary indicators.

Cryotherapy Dataset * **Binary Mapping:** Converts 'sex' to a binary format and adjusts column naming similarly to the Immunotherapy dataset to maintain consistency. * **ne-Hot Encoding:** The 'Type' column is treated with one-hot encoding to convert categorical treatment types into a series of binary columns, which are more suitable for machine learning models. * **Concatenation of One-Hot Encoded Data:** These binary columns are merged back into the primary dataset, ensuring that all data remains integrated and accessible for subsequent analysis.

1.3.2 MODELING AND HYERPARAMETER OPTIMIZATION

In aligning with the methods of the replicated study, we employ the same machine learning models, each optimized through a systematic hyperparameter tuning process. The models and their respective hyperparameters are as follows:

- **Random Forest (RF):** Optimized for **n_estimators** (50, 100, 200), **criterion** ('gini', 'entropy'), **max_depth** (None, 10, 20), **min_samples_split** (2, 5, 10), **min_samples_leaf** (1, 2, 4), **max_features** ('sqrt', 'log2').

- **K-Nearest Neighbors (KNN)**: Optimized for **n_neighbors** (3, 5, 7), **weights** ('uniform', 'distance'), **algorithm** ('auto', 'ball_tree', 'kd_tree', 'brute'), **p** (1, 2).
- **Support Vector Machine (SVM):** Optimized for **C** (0.1, 1, 10), **kernel** ('poly', 'rbf'), **degree** (2, 3).
- **Decision Tree (DT):** Optimized for **criterion** ('gini', 'entropy'), **splitter** ('best', 'random'), **max_depth** (None, 10, 20), **min_samples_split** (2, 5, 10), **min_samples_leaf** (1, 2, 4), **max_features** ('sqrt', 'log2').

Hyperparameter tuning is conducted using **GridSearchCV**, which incorporates a **5-fold cross-validation** to ensure robustness and reliability in model performance. This meticulous optimization process ensures that each model is finely tuned to provide the best possible accuracy and generalizability on diverse medical datasets.

1.3.3 PERFORMANCE METRICS

In alignment with the replicated study from the Asian Journal of Computer and Information Systems, we utilize the same performance metrics to evaluate the effectiveness of the machine learning algorithms implemented in our research. Each model was applied to the designated healthcare datasets, and their performance was quantitatively assessed using the following metrics:

- **Training Accuracy:** This metric indicates the proportion of correct predictions made by the model on the training dataset. It provides insight into how well the model learns from the data, mirroring the approach taken in the replicated study.
- **Testing Accuracy:** Reflects the proportion of correct predictions on the testing dataset, crucial for understanding how well the model generalizes to new, unseen data.
- **Training Time:** Measures the duration of time the model takes to learn from the training data, highlighting the computational efficiency of the model during the training phase.
- **Testing Time:** The time required by the model to make predictions on the testing dataset, indicative of the model's operational efficiency.

Additionally, to further enhance our evaluation framework, we have incorporated the measurement of the **Training and Testing F1 Score**, metrics not used in the replicated study. The F1 Score is a harmonic mean of precision and recall, providing a more nuanced view of model performance, particularly in the context of imbalanced datasets:

- **Training F1 Score:** This metric helps assess the balance between precision and recall during the model's training phase, offering deeper insights into the model's predictive performance and accuracy.
- **Testing F1 Score:** Evaluates the model's precision and recall on the testing dataset, which is crucial for applications where the cost of false positives and false negatives is significant.

By utilizing these metrics, which were also employed in the original study, we ensure a consistent and comparative approach to evaluating model performance.

1.4 RESULTS

```
[ ]: # Data uploading, preprocessing & visualization
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
```

```
[ ]: data = pd.read_csv('results.csv')
```

```
[ ]: data
```

```
[ ]:                                  dataset model  train_time  eval_time  \
     0                           breast_cancer    rf    0.183281   0.027984
     1                           breast_cancer   knn    0.002406   0.018223
     2                           breast_cancer   svm    0.029788   0.012569
     3                           breast_cancer    dt    0.017596   0.005135
     4     chronic_kidney_disease_mean_imputed    rf    0.269601   0.044630
     5     chronic_kidney_disease_mean_imputed   knn    0.005406   0.024386
     6     chronic_kidney_disease_mean_imputed   svm    0.010864   0.006508
     7     chronic_kidney_disease_mean_imputed    dt    0.004596   0.001432
     8   chronic_kidney_disease_median_imputed    rf    0.191878   0.040303
     9   chronic_kidney_disease_median_imputed   knn    0.005036   0.017453
     10  chronic_kidney_disease_median_imputed   svm    0.010226   0.003573
     11  chronic_kidney_disease_median_imputed    dt    0.003194   0.001441
     12  chronic_kidney_disease_missing_dropped    rf    0.273432   0.035553
     13  chronic_kidney_disease_missing_dropped   knn    0.004863   0.015425
     14  chronic_kidney_disease_missing_dropped   svm    0.009367   0.004073
     15  chronic_kidney_disease_missing_dropped    dt    0.008460   0.003340
     16                               covid_19    rf    0.194533   0.030823
     17                               covid_19   knn    0.003052   0.031629
     18                               covid_19   svm    0.007811   0.003878
     19                               covid_19    dt    0.007249   0.006057
     20                            cryotherapy    rf    0.219233   0.040224
     21                            cryotherapy   knn    0.007742   0.034397
     22                            cryotherapy   svm    0.007215   0.003353
     23                            cryotherapy    dt    0.003139   0.002361
     24                 hepatitis_mean_imputed    rf    0.237288   0.038766
     25                 hepatitis_mean_imputed   knn    0.006000   0.018121
     26                 hepatitis_mean_imputed   svm    0.012891   0.005145
     27                 hepatitis_mean_imputed    dt    0.009227   0.006996
     28               hepatitis_median_imputed    rf    0.376774   0.041732
     29               hepatitis_median_imputed   knn    0.005754   0.022186
     30               hepatitis_median_imputed   svm    0.010769   0.006719
     31               hepatitis_median_imputed    dt    0.007189   0.003993
     32               hepatitis_missing_dropped    rf    0.208498   0.039103
     33               hepatitis_missing_dropped   knn    0.005799   0.018653
     34               hepatitis_missing_dropped   svm    0.013068   0.006910
```

```
35              hepatitis_missing_dropped    dt    0.004419   0.002771
36                           immunotherapy   rf    0.318120   0.035794
37                           immunotherapy  knn    0.006032   0.025356
38                           immunotherapy  svm    0.004922   0.003697
39                           immunotherapy   dt    0.005246   0.002195

    train_acc  eval_acc  train_f1   eval_f1
0    1.000000  0.964912  1.000000  0.952381
1    0.949451  0.947368  0.929231  0.926829
2    0.916484  0.947368  0.880503  0.925000
3    0.980220  0.947368  0.973451  0.930233
4    1.000000  1.000000  1.000000  1.000000
5    0.850000  0.700000  0.865922  0.727273
6    0.621875  0.662500  0.743100  0.776860
7    0.971875  0.987500  0.976864  0.990291
8    1.000000  0.987500  1.000000  0.990291
9    0.850000  0.712500  0.865169  0.741573
10   0.618750  0.637500  0.752033  0.771654
11   0.981250  0.987500  0.984615  0.990291
12   1.000000  1.000000  1.000000  1.000000
13   0.849206  0.812500  0.612245  0.500000
14   0.785714  0.812500  0.341463  0.500000
15   1.000000  1.000000  1.000000  1.000000
16   1.000000  0.333333  1.000000  0.166667
17   0.636364  0.333333  0.494949  0.166667
18   1.000000  0.333333  1.000000  0.166667
19   0.636364  0.333333  0.494949  0.166667
20   1.000000  0.777778  1.000000  0.800000
21   0.902778  0.611111  0.909091  0.666667
22   0.875000  0.611111  0.888889  0.666667
23   0.916667  0.611111  0.923077  0.631579
24   1.000000  0.774194  1.000000  0.862745
25   0.862903  0.774194  0.920188  0.867925
26   0.798387  0.774194  0.887892  0.872727
27   0.935484  0.741935  0.959596  0.840000
28   1.000000  0.774194  1.000000  0.857143
29   0.870968  0.774194  0.924528  0.867925
30   0.798387  0.774194  0.887892  0.872727
31   0.927419  0.741935  0.953846  0.826087
32   1.000000  0.937500  1.000000  0.965517
33   0.875000  0.875000  0.928571  0.933333
34   0.843750  0.875000  0.913793  0.933333
35   0.921875  0.875000  0.953271  0.928571
36   0.986111  0.833333  0.991304  0.896552
37   0.833333  0.722222  0.906250  0.827586
38   0.805556  0.722222  0.892308  0.838710
39   0.902778  0.777778  0.936937  0.857143
```

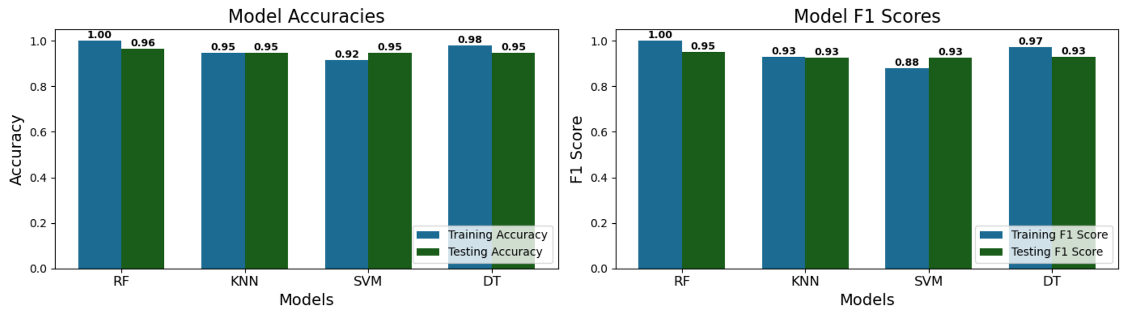
1.4.1 Wisconsin Breast Cancer

Training And Testing Time

Accuracy and F1 Score

```
[ ]: # Filtering the data for the chronic_kidney_disease_mean_imputted dataset
breast_cancer_data = data[data['dataset'] == 'breast_cancer']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(breast_cancer_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, breast_cancer_data['train_acc'], width, label='Training Accuracy',␣
 ↪color=colors['TRAIN'])
ax1.bar([p + width for p in x], breast_cancer_data['eval_acc'], width,␣
 ↪label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(breast_cancer_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(breast_cancer_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, breast_cancer_data['train_f1'], width, label='Training F1 Score',␣
 ↪color=colors['TRAIN'])
ax2.bar([p + width for p in x], breast_cancer_data['eval_f1'], width,␣
 ↪label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(breast_cancer_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(breast_cancer_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
```

```
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(breast_cancer_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(breast_cancer_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



1.4.2 Chronic Kidney Disease

Training And Testing Time

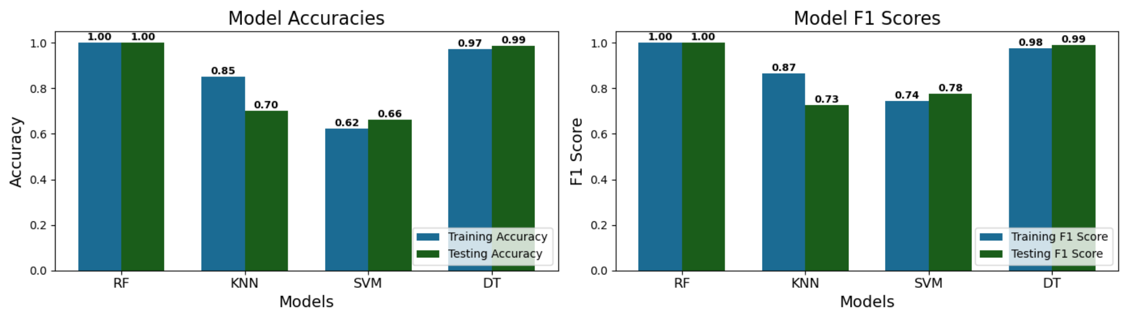
Accuracy and F1 Score

Mean Imputted

```
[ ]: # Filtering the data for the chronic_kidney_disease_mean_imputted dataset
     ckd_mean_imputted_data = data[data['dataset'] ==␣
      ↪'chronic_kidney_disease_mean_imputted']

     # Creating the figure and axes for two histograms side by side again
     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

     # Define the positions for the bars
     x = np.arange(len(ckd_mean_imputted_data))
```

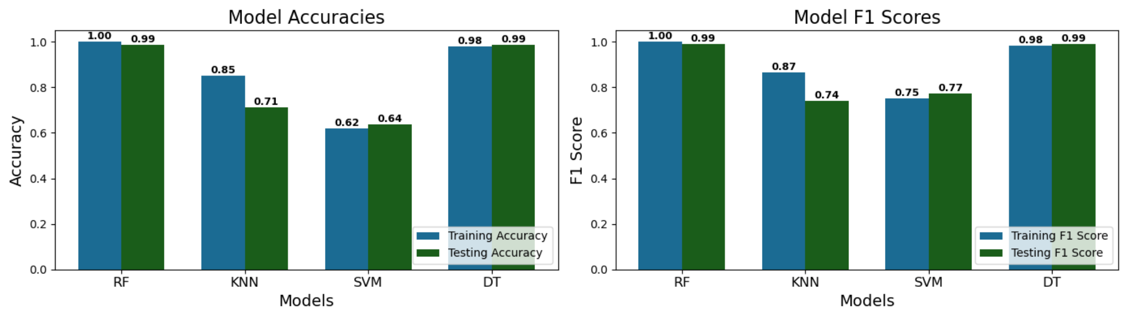
```
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, ckd_mean_imputed_data['train_acc'], width, label='Training␣
↪Accuracy', color=colors['TRAIN'])
ax1.bar([p + width for p in x], ckd_mean_imputed_data['eval_acc'], width,␣
↪label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(ckd_mean_imputed_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(ckd_mean_imputed_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, ckd_mean_imputed_data['train_f1'], width, label='Training F1␣
↪Score', color=colors['TRAIN'])
ax2.bar([p + width for p in x], ckd_mean_imputed_data['eval_f1'], width,␣
↪label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(ckd_mean_imputed_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(ckd_mean_imputed_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(ckd_mean_imputed_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
```

```
ax2.set_xticklabels(ckd_mean_imputted_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



Median Imputted

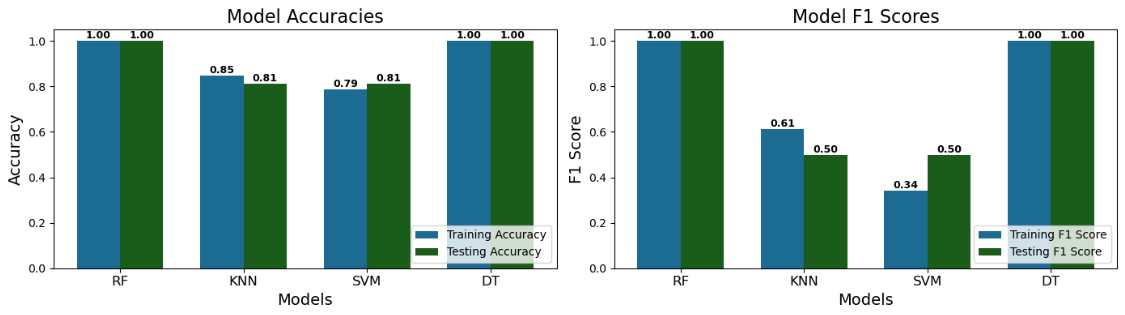
[ ]:
```
# Filtering the data for the chronic_kidney_disease_median_imputted dataset
ckd_median_imputted_data = data[data['dataset'] ==␣
↪'chronic_kidney_disease_median_imputted']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(ckd_median_imputted_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, ckd_median_imputted_data['train_acc'], width, label='Training␣
↪Accuracy', color=colors['TRAIN'])
ax1.bar([p + width for p in x], ckd_median_imputted_data['eval_acc'], width,␣
↪label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(ckd_median_imputted_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(ckd_median_imputted_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, ckd_median_imputted_data['train_f1'], width, label='Training F1␣
↪Score', color=colors['TRAIN'])
ax2.bar([p + width for p in x], ckd_median_imputted_data['eval_f1'], width,␣
↪label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(ckd_median_imputted_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(ckd_median_imputted_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(ckd_median_imputted_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(ckd_median_imputted_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



Missing Dropped

```
[ ]: # Filtering the data for the chronic_kidney_disease_missing_dropped dataset
     ckd_missing_dropped_data = data[data['dataset'] ==␣
      ↪'chronic_kidney_disease_missing_dropped']

     # Creating the figure and axes for two histograms side by side again
     fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

     # Define the positions for the bars
     x = np.arange(len(ckd_missing_dropped_data))
     width = 0.35

     # Colors for the bars
     colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

     # Plotting accuracies
     ax1.bar(x, ckd_missing_dropped_data['train_acc'], width, label='Training␣
      ↪Accuracy', color=colors['TRAIN'])
     ax1.bar([p + width for p in x], ckd_missing_dropped_data['eval_acc'], width,␣
      ↪label='Testing Accuracy', color=colors['TEST'])

     # Adding labels above bars for accuracies
     for i, v in enumerate(ckd_missing_dropped_data['train_acc']):
         ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
      ↪color='black', fontsize=9)
     for i, v in enumerate(ckd_missing_dropped_data['eval_acc']):
         ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
      ↪fontweight='bold', color='black', fontsize=9)

     # Plotting F1 scores
     ax2.bar(x, ckd_missing_dropped_data['train_f1'], width, label='Training F1␣
      ↪Score', color=colors['TRAIN'])
     ax2.bar([p + width for p in x], ckd_missing_dropped_data['eval_f1'], width,␣
      ↪label='Testing F1 Score', color=colors['TEST'])

     # Adding labels above bars for F1 scores
     for i, v in enumerate(ckd_missing_dropped_data['train_f1']):
         ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
      ↪color='black', fontsize=9)
     for i, v in enumerate(ckd_missing_dropped_data['eval_f1']):
         ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
      ↪fontweight='bold', color='black', fontsize=9)

     # Styling and labeling
```

```
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(ckd_missing_dropped_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(ckd_missing_dropped_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



1.4.3 Covid 19

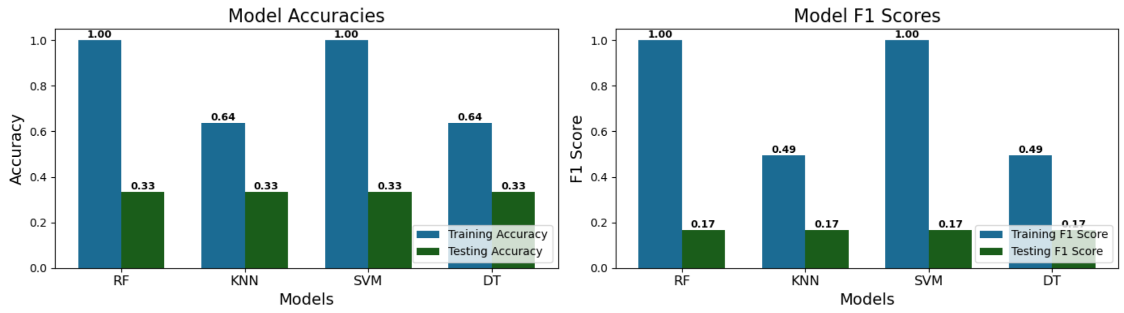
Training And Testing Time

Accuracy and F1 Score

[ ]:
```
# Filtering the data for the covid_19 dataset
covid_19_data = data[data['dataset'] == 'covid_19']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(covid_19_data))
width = 0.35

# Colors for the bars
```

```python
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, covid_19_data['train_acc'], width, label='Training Accuracy',␣
 ↪color=colors['TRAIN'])
ax1.bar([p + width for p in x], covid_19_data['eval_acc'], width,␣
 ↪label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(covid_19_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(covid_19_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, covid_19_data['train_f1'], width, label='Training F1 Score',␣
 ↪color=colors['TRAIN'])
ax2.bar([p + width for p in x], covid_19_data['eval_f1'], width, label='Testing␣
 ↪F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(covid_19_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(covid_19_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(covid_19_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(covid_19_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)
```

```
plt.tight_layout()
plt.show()
```



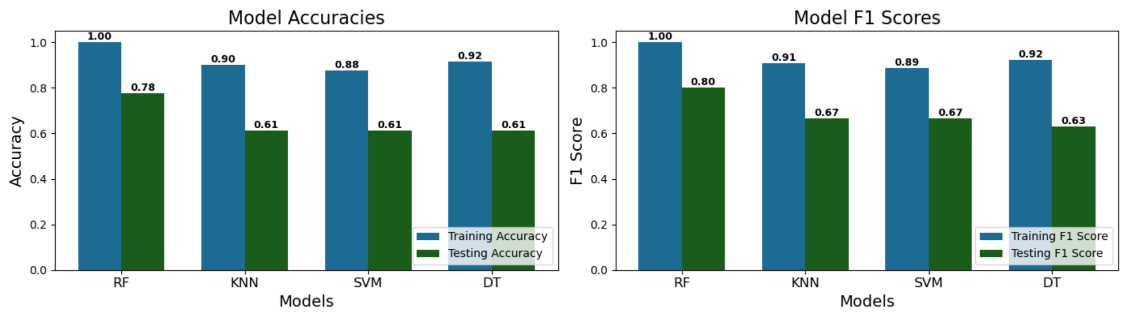
1.4.4 Cryotherapy

Training And Testing Time

Accuracy and F1 Score

[ ]:
```
# Filtering the data for the cryotherapy dataset
cryotherapy_data = data[data['dataset'] == 'cryotherapy']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(cryotherapy_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, cryotherapy_data['train_acc'], width, label='Training Accuracy',␣
 ↪color=colors['TRAIN'])
ax1.bar([p + width for p in x], cryotherapy_data['eval_acc'], width,␣
 ↪label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(cryotherapy_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(cryotherapy_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)
```

```
# Plotting F1 scores
ax2.bar(x, cryotherapy_data['train_f1'], width, label='Training F1 Score',␣
 ↪color=colors['TRAIN'])
ax2.bar([p + width for p in x], cryotherapy_data['eval_f1'], width,␣
 ↪label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(cryotherapy_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(cryotherapy_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(cryotherapy_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(cryotherapy_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



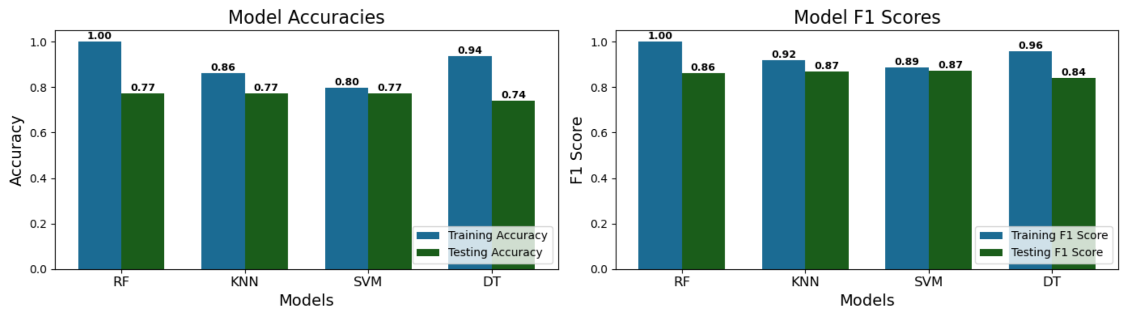
1.4.5 Hepatisis

Training And Testing Time

Accuracy and F1 Score

Mean Imputted

[ ]:
```
# Filtering the data for the hepatitis_mean_imputted dataset
hepatitis_mean_imputted_data = data[data['dataset'] ==␣
↪'hepatitis_mean_imputted']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(hepatitis_mean_imputted_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, hepatitis_mean_imputted_data['train_acc'], width, label='Training␣
↪Accuracy', color=colors['TRAIN'])
ax1.bar([p + width for p in x], hepatitis_mean_imputted_data['eval_acc'],␣
↪width, label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(hepatitis_mean_imputted_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(hepatitis_mean_imputted_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, hepatitis_mean_imputted_data['train_f1'], width, label='Training F1␣
↪Score', color=colors['TRAIN'])
ax2.bar([p + width for p in x], hepatitis_mean_imputted_data['eval_f1'], width,␣
↪label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(hepatitis_mean_imputted_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(hepatitis_mean_imputted_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
  ↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(hepatitis_mean_imputed_data['model'].str.upper(),␣
  ↪fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(hepatitis_mean_imputed_data['model'].str.upper(),␣
  ↪fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```

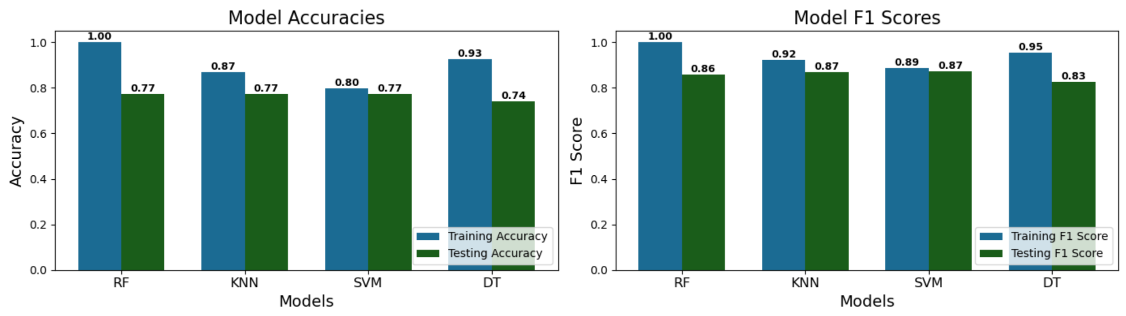


Median Imputted

[ ]:
```
# Filtering the data for the hepatitis_median_imputted dataset
hepatitis_median_imputted_data = data[data['dataset'] ==␣
  ↪'hepatitis_median_imputted']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
```

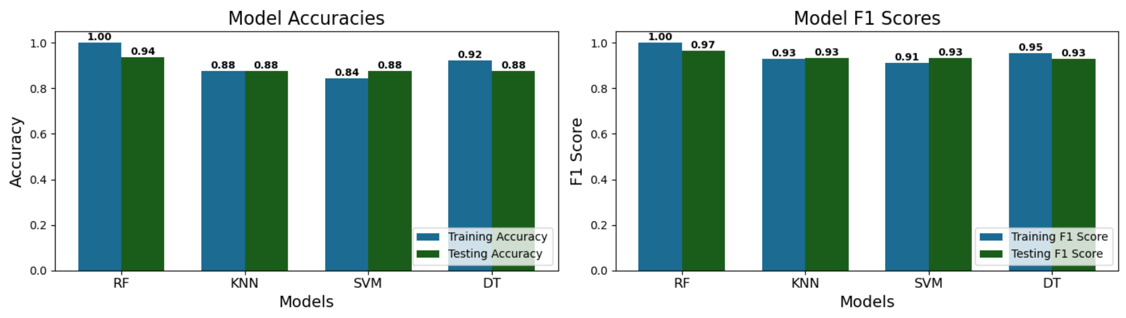
```
x = np.arange(len(hepatitis_median_imputted_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, hepatitis_median_imputted_data['train_acc'], width, label='Training␣
 ↪Accuracy', color=colors['TRAIN'])
ax1.bar([p + width for p in x], hepatitis_median_imputted_data['eval_acc'],␣
 ↪width, label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(hepatitis_median_imputted_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(hepatitis_median_imputted_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, hepatitis_median_imputted_data['train_f1'], width, label='Training␣
 ↪F1 Score', color=colors['TRAIN'])
ax2.bar([p + width for p in x], hepatitis_median_imputted_data['eval_f1'],␣
 ↪width, label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(hepatitis_median_imputted_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(hepatitis_median_imputted_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(hepatitis_median_imputted_data['model'].str.upper(),␣
 ↪fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
```

```
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(hepatitis_median_imputted_data['model'].str.upper(),␣
 ↪fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



Missing Dropped

```
[ ]: # Filtering the data for the hepatitis_missing_dropped dataset
hepatitis_missing_dropped_data = data[data['dataset'] ==␣
 ↪'hepatitis_missing_dropped']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(hepatitis_missing_dropped_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, hepatitis_missing_dropped_data['train_acc'], width, label='Training␣
 ↪Accuracy', color=colors['TRAIN'])
ax1.bar([p + width for p in x], hepatitis_missing_dropped_data['eval_acc'],␣
 ↪width, label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(hepatitis_missing_dropped_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(hepatitis_missing_dropped_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, hepatitis_missing_dropped_data['train_f1'], width, label='Training␣
↪F1 Score', color=colors['TRAIN'])
ax2.bar([p + width for p in x], hepatitis_missing_dropped_data['eval_f1'],␣
↪width, label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(hepatitis_missing_dropped_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
↪color='black', fontsize=9)
for i, v in enumerate(hepatitis_missing_dropped_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(hepatitis_missing_dropped_data['model'].str.upper(),␣
↪fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(hepatitis_missing_dropped_data['model'].str.upper(),␣
↪fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



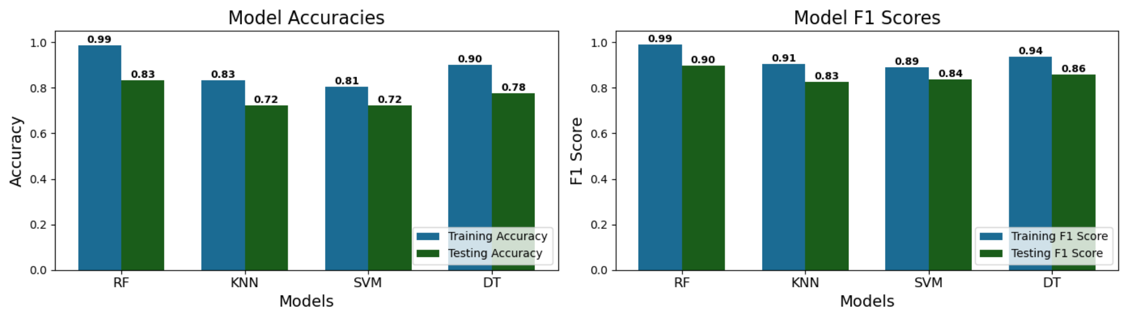
1.4.6 Immunotherapy

Training And Testing Time

Accuracy and F1 Score

```
[ ]: # Filtering the data for the immunotherapy dataset
immunotherapy_data = data[data['dataset'] == 'immunotherapy']

# Creating the figure and axes for two histograms side by side again
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 4))

# Define the positions for the bars
x = np.arange(len(immunotherapy_data))
width = 0.35

# Colors for the bars
colors = {'TRAIN': '#1B6B93', 'TEST': '#1A5D1A'}

# Plotting accuracies
ax1.bar(x, immunotherapy_data['train_acc'], width, label='Training Accuracy',␣
 ↪color=colors['TRAIN'])
ax1.bar([p + width for p in x], immunotherapy_data['eval_acc'], width,␣
 ↪label='Testing Accuracy', color=colors['TEST'])

# Adding labels above bars for accuracies
for i, v in enumerate(immunotherapy_data['train_acc']):
    ax1.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(immunotherapy_data['eval_acc']):
    ax1.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Plotting F1 scores
ax2.bar(x, immunotherapy_data['train_f1'], width, label='Training F1 Score',␣
 ↪color=colors['TRAIN'])
```

```
ax2.bar([p + width for p in x], immunotherapy_data['eval_f1'], width,␣
 ↪label='Testing F1 Score', color=colors['TEST'])

# Adding labels above bars for F1 scores
for i, v in enumerate(immunotherapy_data['train_f1']):
    ax2.text(i, v, f"{v:.2f}", ha='center', va='bottom', fontweight='bold',␣
 ↪color='black', fontsize=9)
for i, v in enumerate(immunotherapy_data['eval_f1']):
    ax2.text(i + width, v, f"{v:.2f}", ha='center', va='bottom',␣
 ↪fontweight='bold', color='black', fontsize=9)

# Styling and labeling
ax1.set_xlabel('Models', fontsize=14)
ax1.set_ylabel('Accuracy', fontsize=14)
ax1.set_title('Model Accuracies', fontsize=16)
ax1.set_xticks(x + width / 2)
ax1.set_xticklabels(immunotherapy_data['model'].str.upper(), fontsize=12)
ax1.legend(loc='lower right')
ax1.set_ylim(0.0, 1.05)

ax2.set_xlabel('Models', fontsize=14)
ax2.set_ylabel('F1 Score', fontsize=14)
ax2.set_title('Model F1 Scores', fontsize=16)
ax2.set_xticks(x + width / 2)
ax2.set_xticklabels(immunotherapy_data['model'].str.upper(), fontsize=12)
ax2.legend(loc='lower right')
ax2.set_ylim(0.0, 1.05)

plt.tight_layout()
plt.show()
```



1.5 COMPARISON

In our replication of the study, we follow the original approach of categorizing model performance into four levels: **excellent execution, good execution, fair execution, and inadequate**

execution. However, the original study does not specify the metrics or methodology used to classify these performance categories. In the absence of this information, we have structured our comparison based on a combination of quantitative measures: **Testing Accuracy, Testing F1 Score, and Testing Time.**

In our study, we primarily use **Testing Accuracy** to compare the performance of models across various datasets. If models show identical testing accuracy, we then assess the **Testing F1 Score** for a deeper evaluation of model precision and recall. In cases where the F1 scores are also equivalent, **Testing Time** serves as the final criterion to determine the model's efficiency.

Our Results Comparison

Replicated Study Comparison

Our results in the replicated study exhibit significant differences across almost all datasets when compared to the original study. This variation can primarily be attributed to the lack of detailed methodology disclosed in the original study regarding the specific metrics, hyperparameters, and comparison criteria used. Without this information, our replication had to rely on standard performance metrics (Testing Accuracy, F1 Score, and Testing Time) and commonly used hyperparameters for each model, which likely led to the observed discrepancies in model performance classification across the datasets. This highlights the importance of transparency and detailed documentation in research to ensure reproducibility and accurate comparison of results.