AI Summer 2024

Feature Subset Selection

Through GA-FSS & PSO-FSS in Classification Tasks

MUHAMMAD UMER KHAN

Intro

- Feature subset selection is one of the key problems in pattern recognition and machine learning. [1]
- It is the process of identifying and selecting a subset of relevant features (variables, predictors) for use in model construction.
- Reduces **Overfitting**
- Improves **Accuracy**
- Decreases **Computational Cost**
- Allows **Interpretability**

[1] Syed Imran Ali, Waseem Shahzad “A Feature Subset Selection Method based on Symmetric Uncertainty and Ant Colony Optimization” *International Journal of Computer Applications*

Problem Setup

- **Task:** Classification
- **Baseline:** Decision Tree
- **Datasets**
  - Alzeihmer's Disease Data [3]
  - Weather Classification Data [4]
  - Other Benchmark Datasets

- **Feature Subset Selection**
  1. **Genetic Algorithm**
  2. **Particle Swarm Optimization**
  3. **Particle Swarm Optimization** (Novel initialisation [5])

[3] https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset
[4] https://www.kaggle.com/datasets/nikhil7280/weather-type-classification
[5] Bing Xue∗, Mengjie Zhang, Will N. Browne "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms" *Applied Soft Computing*

Datasets

1. Weather Classification

This dataset is synthetically generated to mimic weather data for classification tasks. It includes various weather-related features and categorizes the weather into four types: Rainy, Sunny, Cloudy, and Snowy. This dataset is designed for practicing classification algorithms, data preprocessing, and outlier detection methods.

Key Features

- **Rows**: 13,200
- **Number of Features**: 10
- **Target**: Weather type (categorical)
- **Preprocessing:** Handling missing values, normalization, and splitting into training and test sets.

	Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Location	Weather Type
0	14.0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	inland	Rainy
1	39.0	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	inland	Cloudy
2	30.0	64	7.0	16.0	clear	1018.72	5	Spring	5.5	mountain	Sunny
3	38.0	83	1.5	82.0	clear	1026.25	7	Spring	1.0	coastal	Sunny
4	27.0	74	17.0	66.0	overcast	990.67	1	Winter	2.5	mountain	Rainy

2. Alzeimer’s Disease Detection

This dataset contains extensive health information for 2,149 patients, each uniquely identified with IDs ranging from 4751 to 6900. The dataset includes demographic details, lifestyle factors, medical history, clinical measurements, cognitive and functional assessments, symptoms, and a diagnosis of Alzheimer's Disease. The data is ideal for researchers and data scientists looking to explore factors associated with Alzheimer's, develop predictive models, and conduct statistical analyses.

Key Steps

- **Rows**: 2149
- **Number of Features**: 33
- **Target**: Diagnosis (binary categorical)
- **Preprocessing:** Handling missing values, normalization, and splitting into training and test sets.

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	...	Forgetfulness	Diagnosis
0	4751	73	0	0	2	22.927749	0	13.297218	6.327112	1.347214	...	0	0
1	4752	89	0	0	0	26.827681	0	4.542524	7.619885	0.518767	...	1	0
2	4753	73	0	3	1	17.795882	0	19.555085	7.844988	1.826335	...	0	0
3	4754	74	1	0	1	33.800817	1	12.209266	8.428001	7.435604	...	0	0
4	4755	89	0	0	0	20.716974	0	18.454356	6.310461	0.795498	...	0	0

Results

Random Forest

Base implementation from Sci-Kit Learn

Weather Classification
Baseline: 90.15%

Alzeimer’s Disease
Baseline: 93.02%

GA-FSS

Implemented GA through the **deap** library
Population: 50
Generations: 20
Crossover: 50%
Mutation: 20%

Weather Classification
Baseline: 90.15% (10 Features)
GA-FSS: 90.78% (8 Features)

Fitness

```
# Predict and calculate accuracy
y_pred = dt.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)

# Calculate Information Gain
mutual_info = mutual_info_classif(X_train, y_train)
relevancy = np.sum(mutual_info[selected_features]) / len(selected_features)

# Fitness function combines accuracy and relevancy
fitness = accuracy + relevancy
```

Alzeimer's Disease
Baseline: 93.02% (33 Features)
GA-FSS: 93.95% (18 Features)

PSO-FSS

Implemented PSO through the **pyswarm** library
Population: 50
Iterations: 20

Fitness

```python
# Predict and calculate accuracy
y_pred = dt.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)

# Calculate Information Gain
mutual_info = mutual_info_classif(X_train, y_train)
relevancy = np.sum(mutual_info[selected_features]) / len(selected_features)

# Fitness function combines accuracy and relevancy
fitness = accuracy + relevancy
```

Weather Classification
Baseline: 90.15% (10 Features)
GA-FSS: 90.78% (8 Features)
PSO-FSS: 90.61% (8 Features)

Alzeimer's Disease
Baseline: 93.02% (33 Features)
GA-FSS: 93.95% (18 Features)
PSO-FSS: 94.11% (22 Features)

Mixed Initialization

- **Traditional Initialisation:**
Each particle is randomly initialised in terms of both the number of features and the combination of individual features.

- **Small Initialisation:**
Initialise each particle using a small number of features, but random combinations of features.

- **Large Initialisation:**
Initialise each particle using a large number of features, but random combinations of features.

- **Mixed initialisation:**
This initialisation strategy combines both the small initialisation and large initialisation strategies.

```python
# Mixed Initialization
def mixed_initialization(swarmsize, num_features):
    particles = []
    for _ in range(swarmsize):
        if random.random() < 0.5:  # 50% chance of small or large initialization
            # Small Initialization
            particle = np.zeros(num_features)
            selected_features = random.sample(range(num_features), k=random.randi
            for idx in selected_features:
                particle[idx] = random.uniform(0, 1)
        else:
            # Large Initialization
            particle = np.ones(num_features)
            selected_features = random.sample(range(num_features), k=random.randi
            for idx in selected_features:
                particle[idx] = random.uniform(0, 1)
        particles.append(particle)
    return np.array(particles)
```

[5] Bing Xue∗, Mengjie Zhang, Will N. Browne “Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms” *Applied Soft Computing*

PSO-FSS Improved

- Custom Implementation
- Mixed Initialization [5]

```python
# Mixed Initialization
def mixed_initialization(swarmsize, num_features):
    particles = []
    for _ in range(swarmsize):
        if random.random() < 0.5:  # 50% chance of small or large initialization
            # Small Initialization
            particle = np.zeros(num_features)
            selected_features = random.sample(range(num_features), k=random.randi
            for idx in selected_features:
                particle[idx] = random.uniform(0, 1)
        else:
            # Large Initialization
            particle = np.ones(num_features)
            selected_features = random.sample(range(num_features), k=random.randi
            for idx in selected_features:
                particle[idx] = random.uniform(0, 1)
        particles.append(particle)
    return np.array(particles)
```

Weather Classification
Baseline: 90.15% (10 Features)
GA-FSS: 90.63% (8 Features)
PSO-FSS: 90.91% (8 Features)
PSO-FSS (MI): 90.78% (8 Features)

Alzeimer's Disease
Baseline: 93.18% (33 Features)
GA-FSS: 93.95% (18 Features)
PSO-FSS: 94.11% (22 Features)
PSO-FSS (MI): 94.26% (30 Features)

[5] Bing Xue∗, Mengjie Zhang, Will N. Browne "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms" *Applied Soft Computing*

Summary

Iris Dataset (Benchmark) [150]

	Method	Accuracy	Features Selected	Computation Time
0	Baseline	1.0000	5	N/A
1	GA	1.0000	2	18.83 seconds
2	PSO	1.0000	2	20.76 seconds
3	Modified PSO	1.0000	3	19.19 seconds

Weather Classification Dataset [13200]

	Method	Accuracy	Features Selected	Computation Time
0	Baseline	0.9015	10	N/A
1	GA	0.9078	8	374.66 seconds
2	PSO	0.9061	8	419.22 seconds
3	Modified PSO	0.9078	8	302.34

Ionosphere Dataset (Benchmark)

	Method	Accuracy	Features Selected	Computation Time
0	Baseline	0.8286	34	N/A
1	GA	0.9524	18	115.08 seconds
2	PSO	0.9524	22	117.99 seconds
3	Modified PSO	0.9619	12	106.16 seconds
1 *	Modified PSO	0.9429	4	114.20 seconds

Alzeimer's Disease Dataset [2149]

	Method	Accuracy	Features Selected	Computation Time
0	Baseline	0.9302	34	N/A
1	GA	0.9395	18	296.54 seconds
2	PSO	0.9411	21	413.94 seconds
3	Modified PSO	0.9426	30	250.40 seconds
1 *	Modified PSO	0.9380	17	205.96 seconds

* Mixed Initialization with a 75% chance of a smaller initial feature subset

Multi Objective Problem

- We can model the problem as a Multi-Objective Problem
- Minimize No. of Features Selected
- Maximize Accuracy

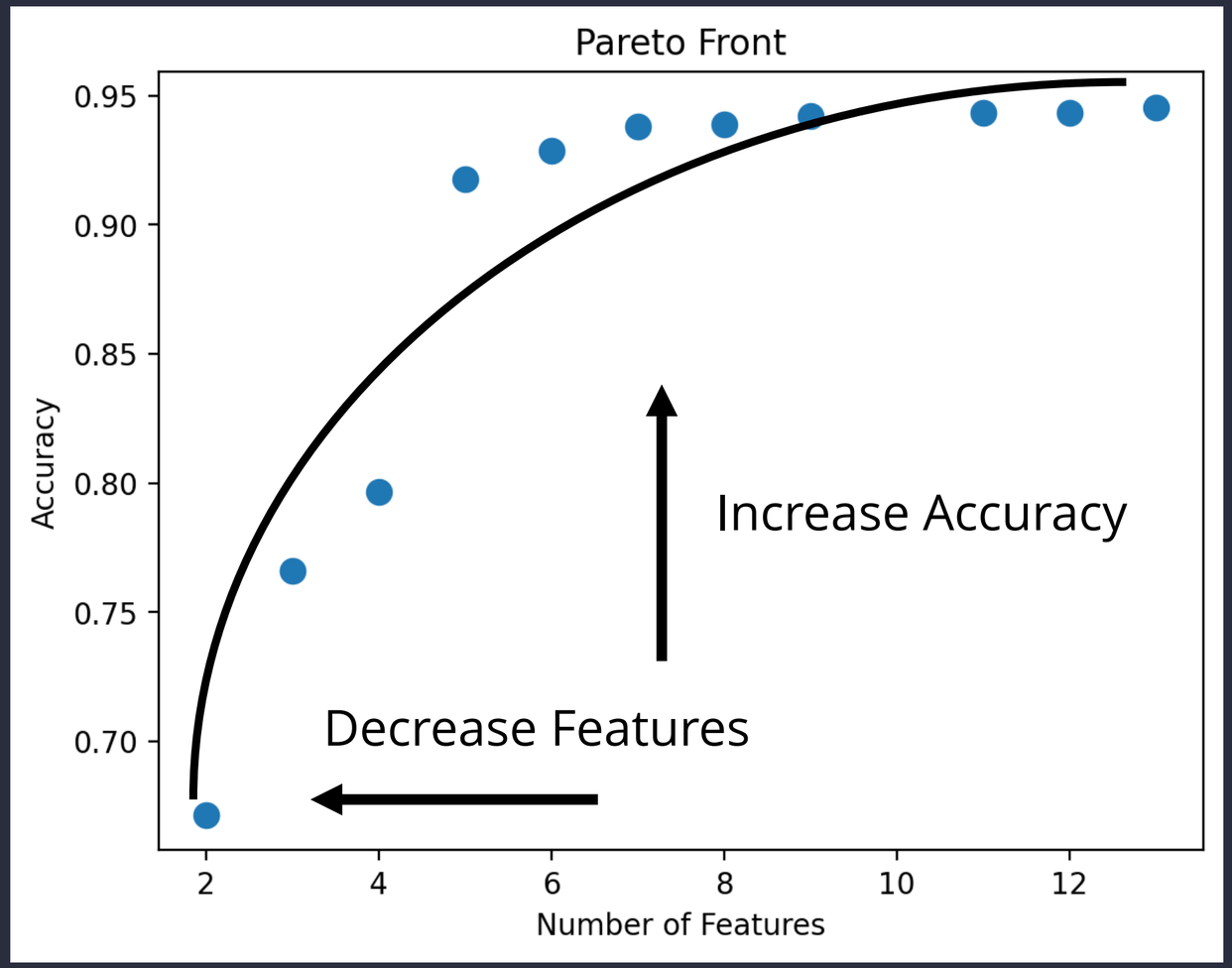
- Intuition:
- We want to reduce the number of subset features we select
- We want to improve accuracy as much as possible
- Previous algorithms only focus on maximizing accuracy

```
def evaluate(individual):
    ...
    return 1 - accuracy, sum(individual)
```

NSGA-II

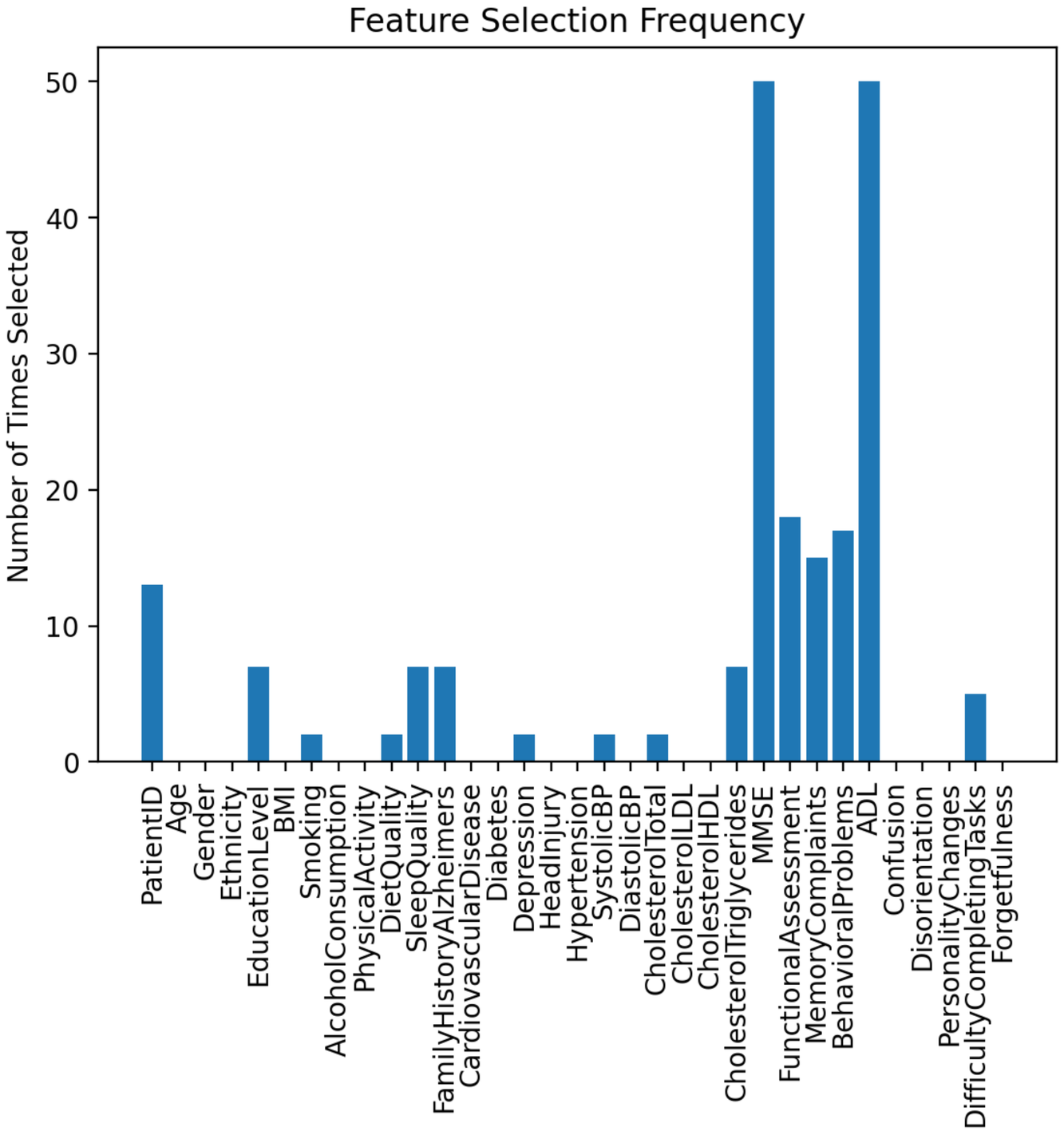
- Implemented through deap library
- Runs faster than all other algorithms
- Test size 30%
- Generations: 50
- Population Size: 50

Best Solution:
Number of Features Selected: 13
Fitness Values (1 - Accuracy, Number of Features):
(0.05452176659841756, 13.0)
Test Accuracy of the Best Solution: 0.937984496124031



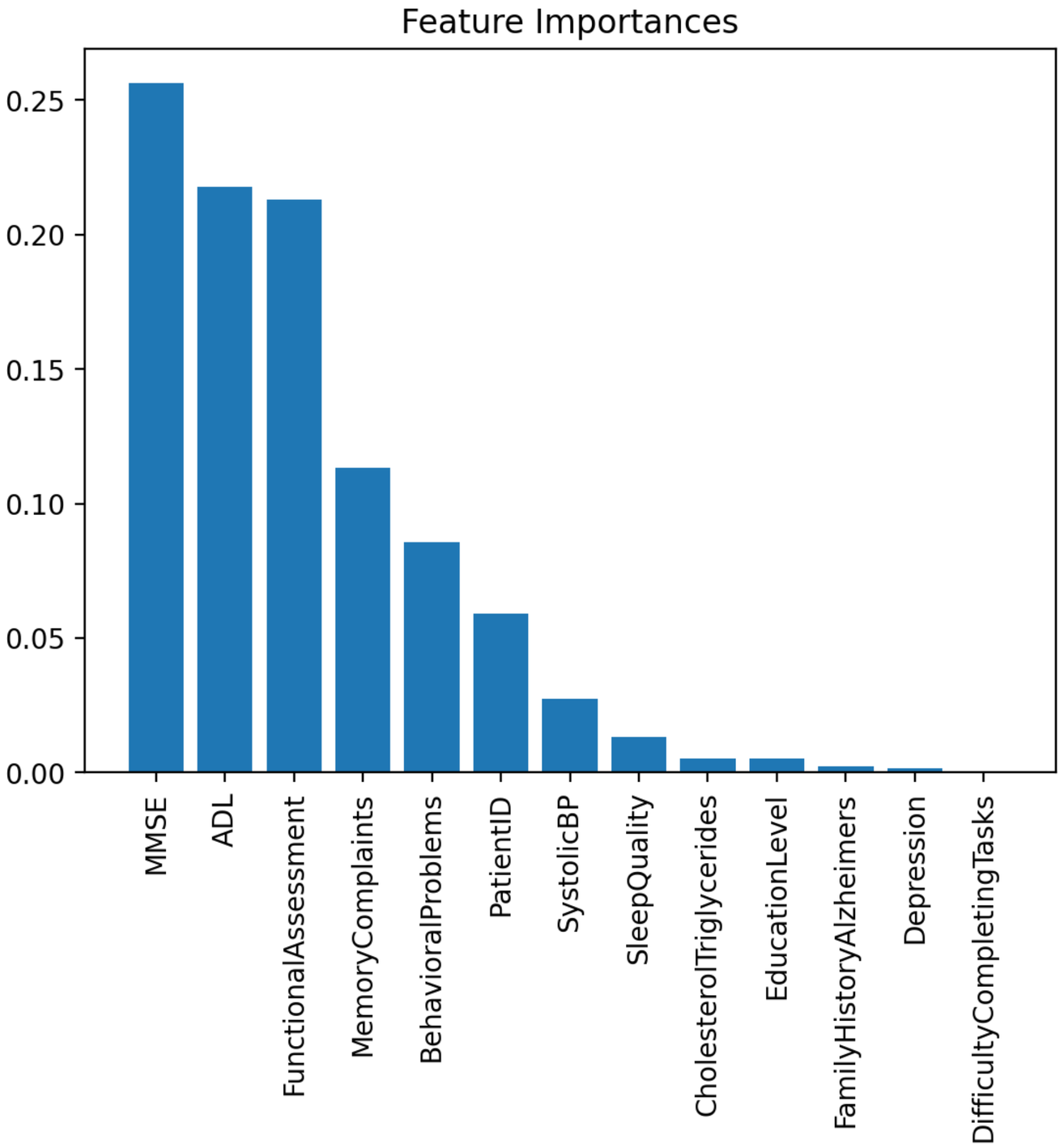
NSGA-II

- Implemented through deap library
- Frequency of Feature Selection Across Pareto-Optimal Solutions
- Ignores features like Age, Gender, Ethnicity, BMI as Alzeimer's



NSGA-II

- Implemented through deap library
- In Decision Trees, feature importance is computed by looking at how much each feature decreases the impurity (e.g., Gini impurity or entropy) of the tree nodes.
- Features that lead to a significant reduction in impurity when they are used in the decision nodes are considered important.
- Importance is often scaled by the total number of nodes where the feature is used and the total number of samples that pass through those nodes.

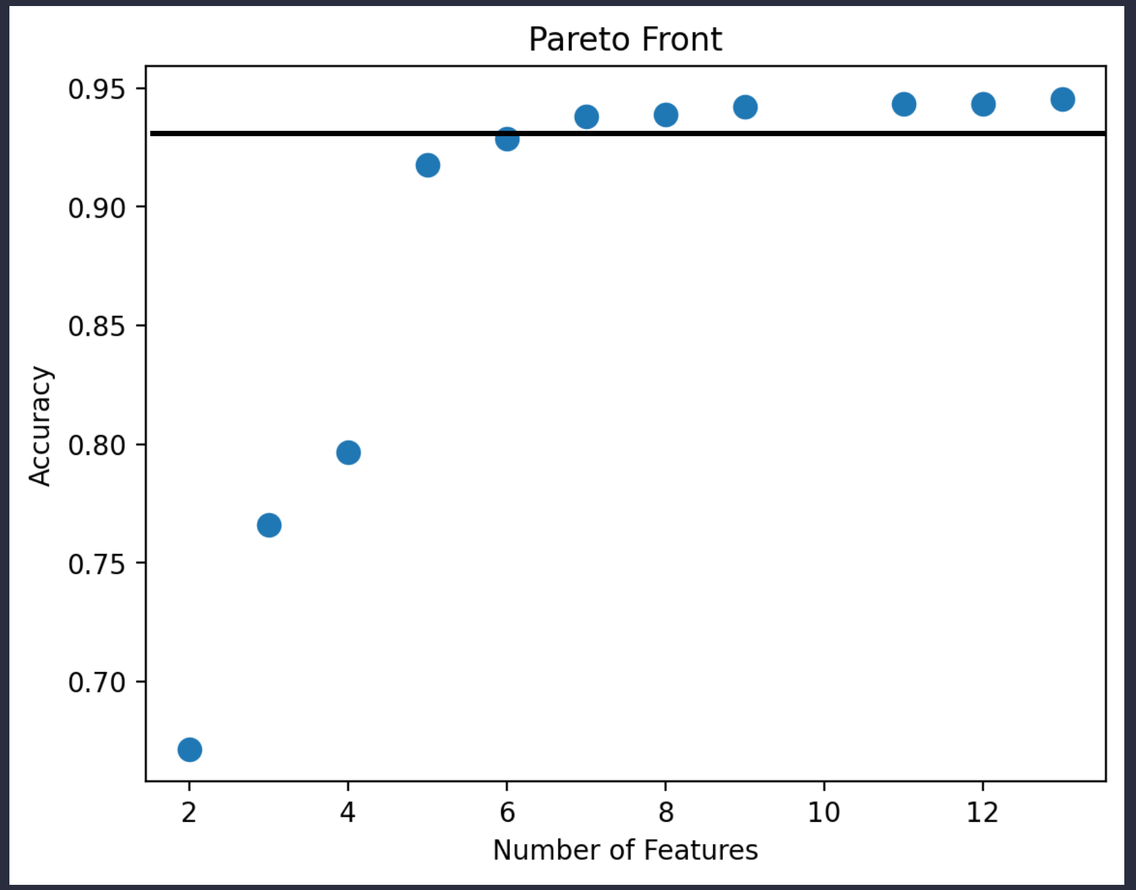


NSGA-II

- Implemented through deap library

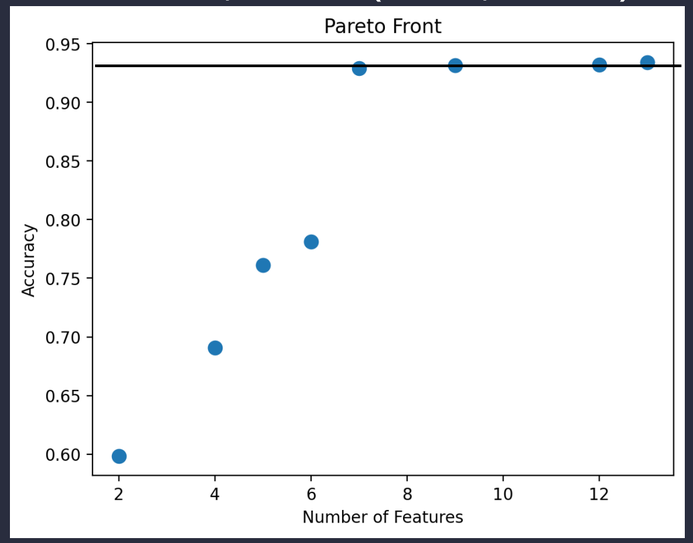
Model	Accuracy	Features
Baseline	93.18	33
GA-FSS	93.95	18
PSO-FSS	94.11	22
PSO-FSS (MI 0.5%)	**94.26***	30
PSO-FSS (MI 0.75%)	93.80	17
NSGA-II	93.79	**13***

Alzeimer’s Disease Detection Dataset

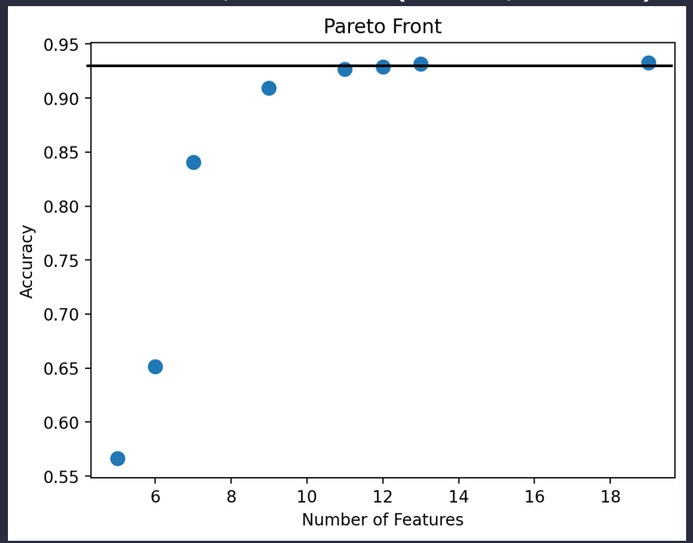


NSGA-II

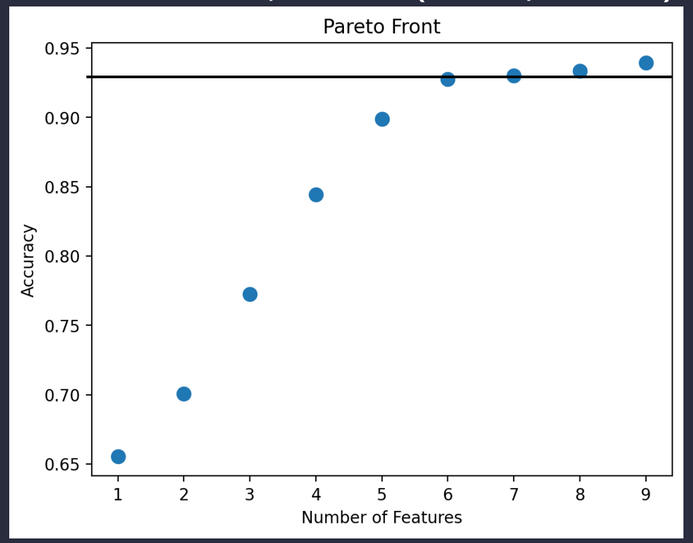
- 20 Gens, N=20 (F=13, 93.64)



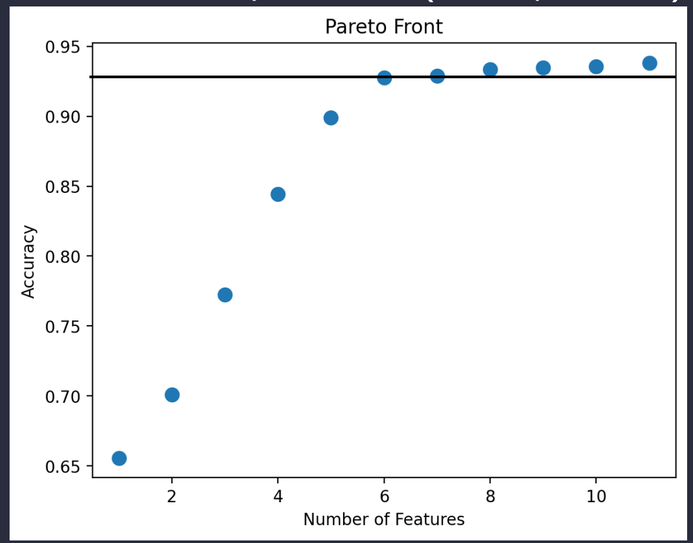
- 20 Gens, N=100 (F=19, 93.80)



- 100 Gens, N=20 (**F=9***, 92.40)

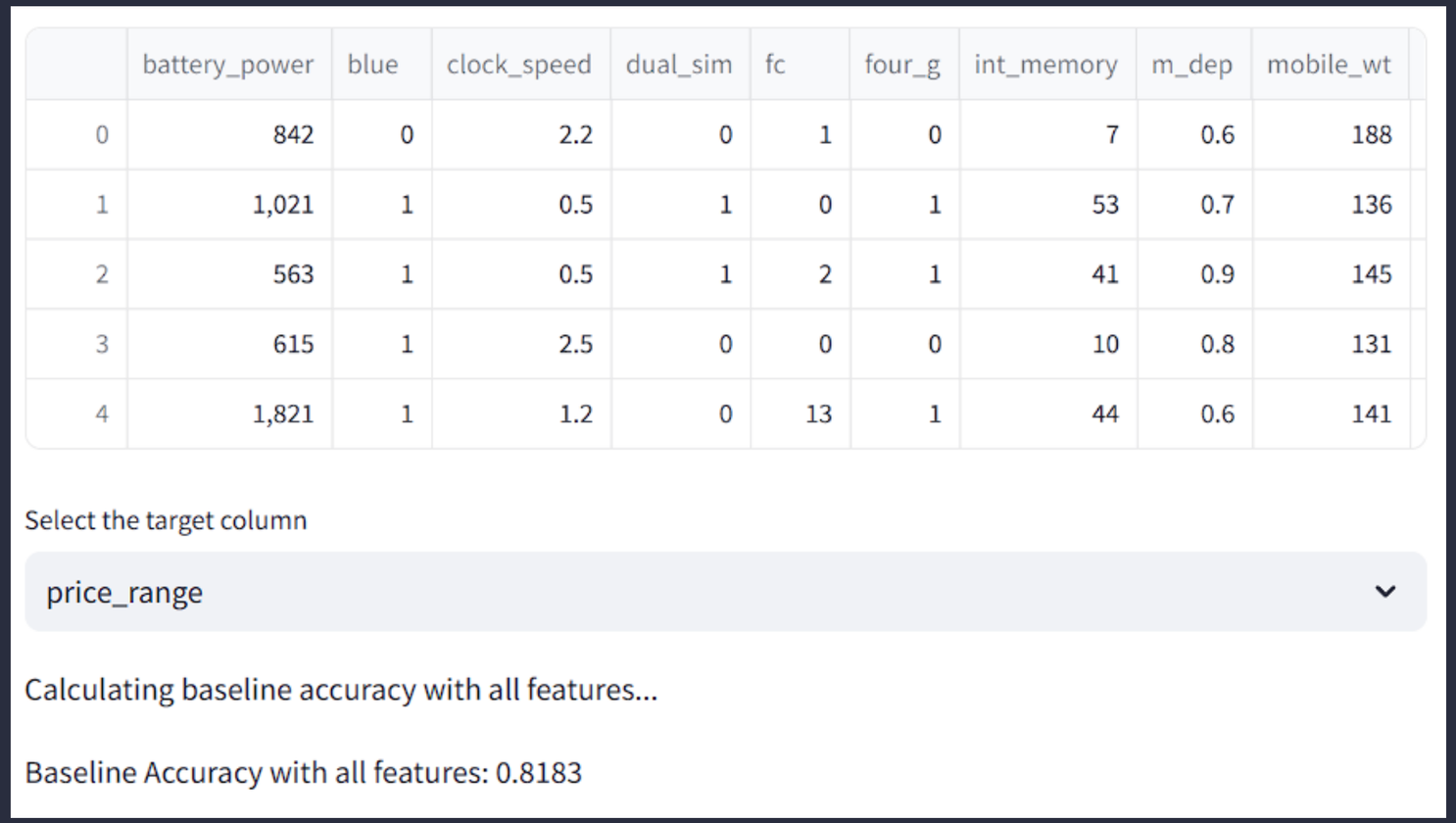


- 100 Gens, N=100 (F=11, 92.40)



Mobile Price Classification

Test on another dataset

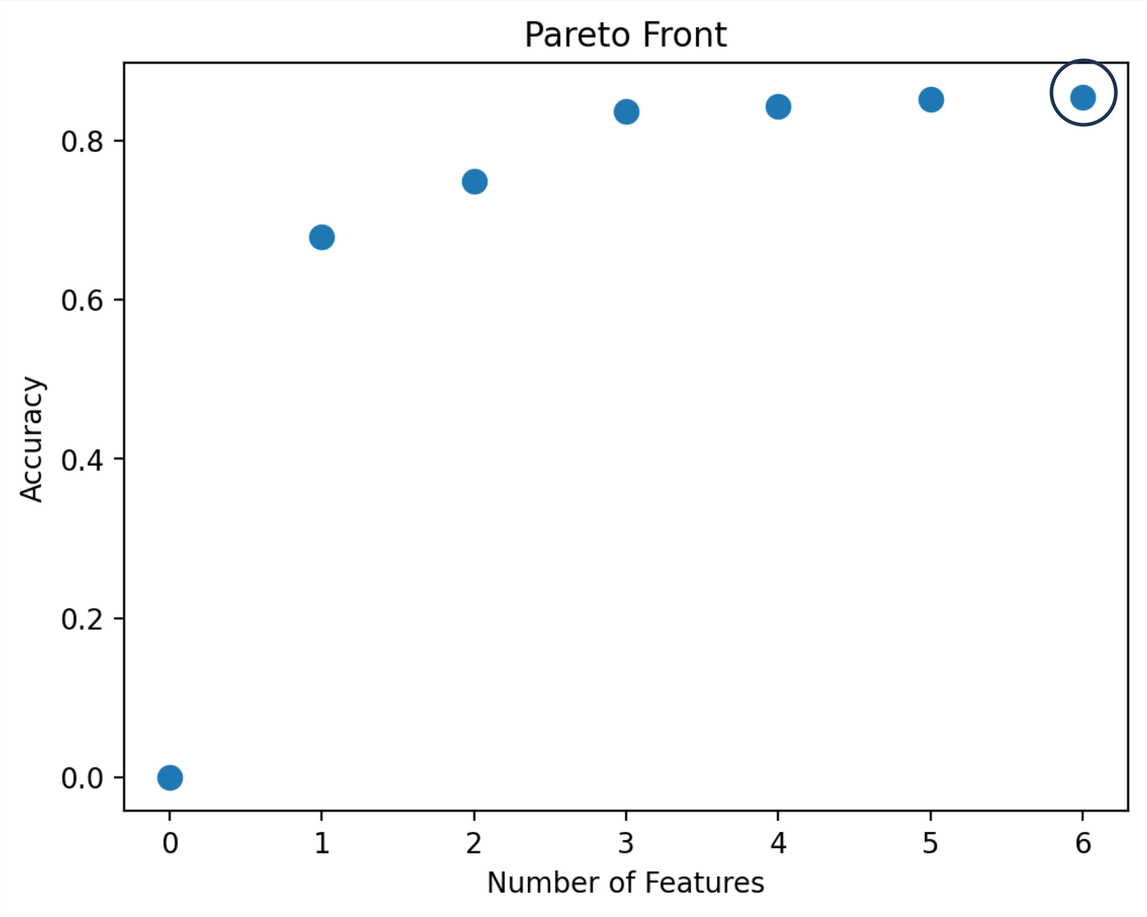


	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt
0	842	0	2.2	0	1	0	7	0.6	188
1	1,021	1	0.5	1	0	1	53	0.7	136
2	563	1	0.5	1	2	1	41	0.9	145
3	615	1	2.5	0	0	0	10	0.8	131
4	1,821	1	1.2	0	13	1	44	0.6	141

Streamlit UI

Mobile Price Classification

Test on another dataset



Selected Features:

	0
0	battery_power
1	fc
2	px_height
3	px_width
4	ram
5	wifi

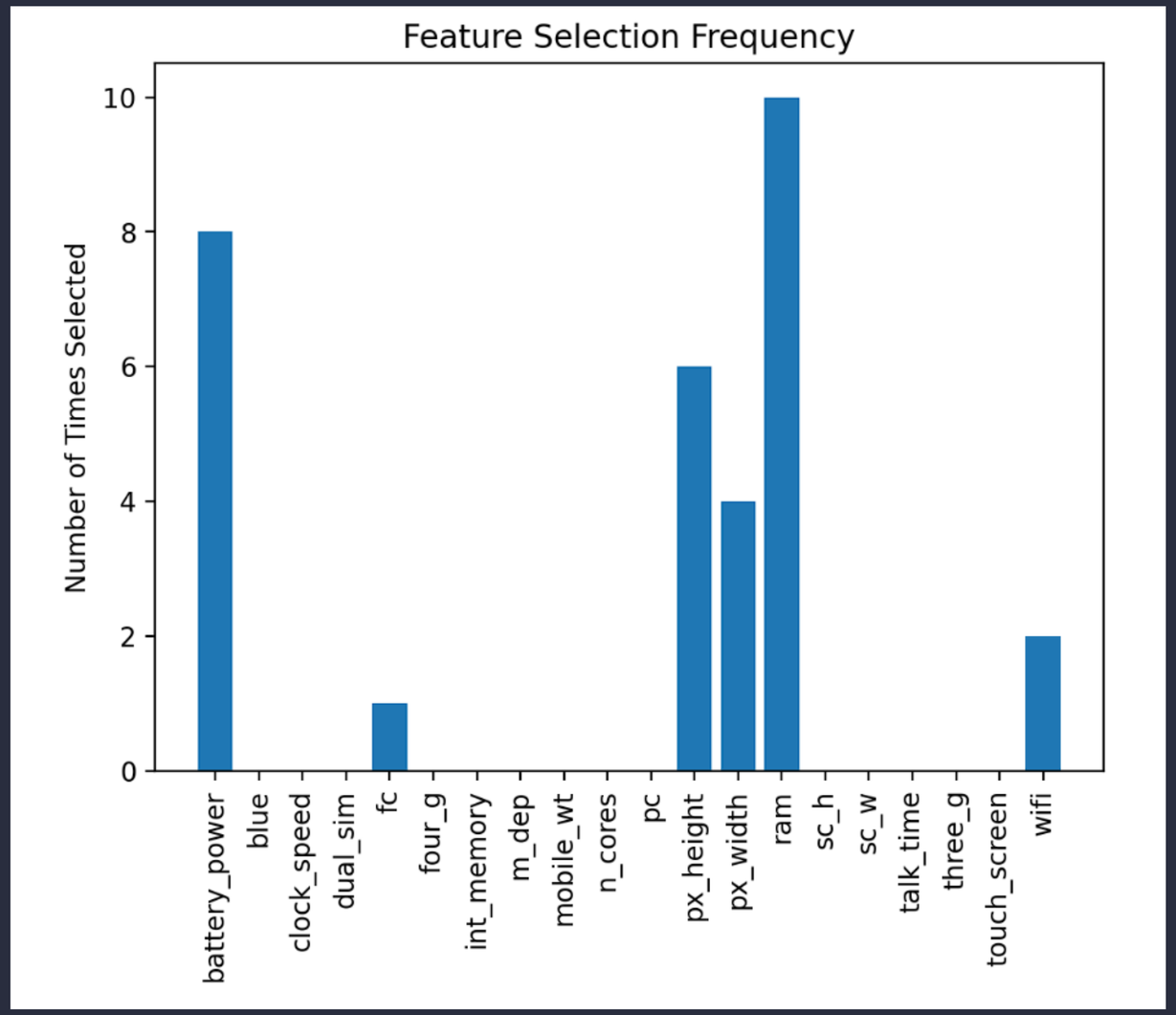
Number of Features Selected: `6`

Fitness Values (1 - Accuracy, Number of Features): `(0.14571428571428569, 6.0)`

Test Accuracy of the Best Solution: `0.835`

Mobile Price Classification

Test on another dataset



Thank You

References to all papers and literature are present and available upon request. Findings in this presentation are based on every reasonable effort to understand existing literature.

Contact: mukhan.bscs21seecs@seecs.edu.pk