

# Determine milk content in coffee

Scientific computing and Machine Learning

MCSN

Prof. Dr. Beate Rhein

TH Cologne

Maleyka Seyidova

# Table of contents:

1. Introduction to the problem
2. Dataset ( image samples)
3. Data preprocessing
4. Network details
5. Results explanation

# Introduction to the problem

---

- Determine milk content in coffee
- Creation of dataset
- Selection of suitable network/algorithm
- Selection of suitable loss
- Performance on test- validation dataset

**Coffee water**



**Coffee littlemilk**



**Coffee more milk**



**Milk only**



# Data preprocessing

---

- **Resizing:** Each class of images are resized to 32x32x3
- **Image array:** Saved in list named `resized_images`
- **Label array:** according to image class 0, 1, 2 and 3
- $(4*225)=900$  - training ;  $(4*75)=300$  – testing

# Dataset details

---

Label	Class	Train samples	Test samples
0	Coffee water	225	75
1	Coffee little milk	225	75
2	Coffee more milk	225	75
3	Milk only	225	75

# Network details : using CNN

---

- Activation function: ReLu
- Optimizer: Adam
- Number of classes: 4
- Network structure: 2 convolutional layers, 1 fully connected layer
- Global variables initialization: weights and biases initialized with normal distribution, std=0.1
- Loss\_function: Softmax cross-entropy loss
- Batch size: 50
- Data validation: 75% for training , 25% for testing

```
1
4
4
4459524
```

```
step 85, test accuracy 1
step 90, training accuracy 1
step 90, test accuracy 1
step 95, training accuracy 1
step 95, test accuracy 1
--- 4.51380395889 seconds ---
total test accuracy 1
```

```
Process finished with exit code 0
```

- Total trainable parameters: 4459524
- Execution time for 300 test samples:  
4.79873800278 seconds

# Results

---

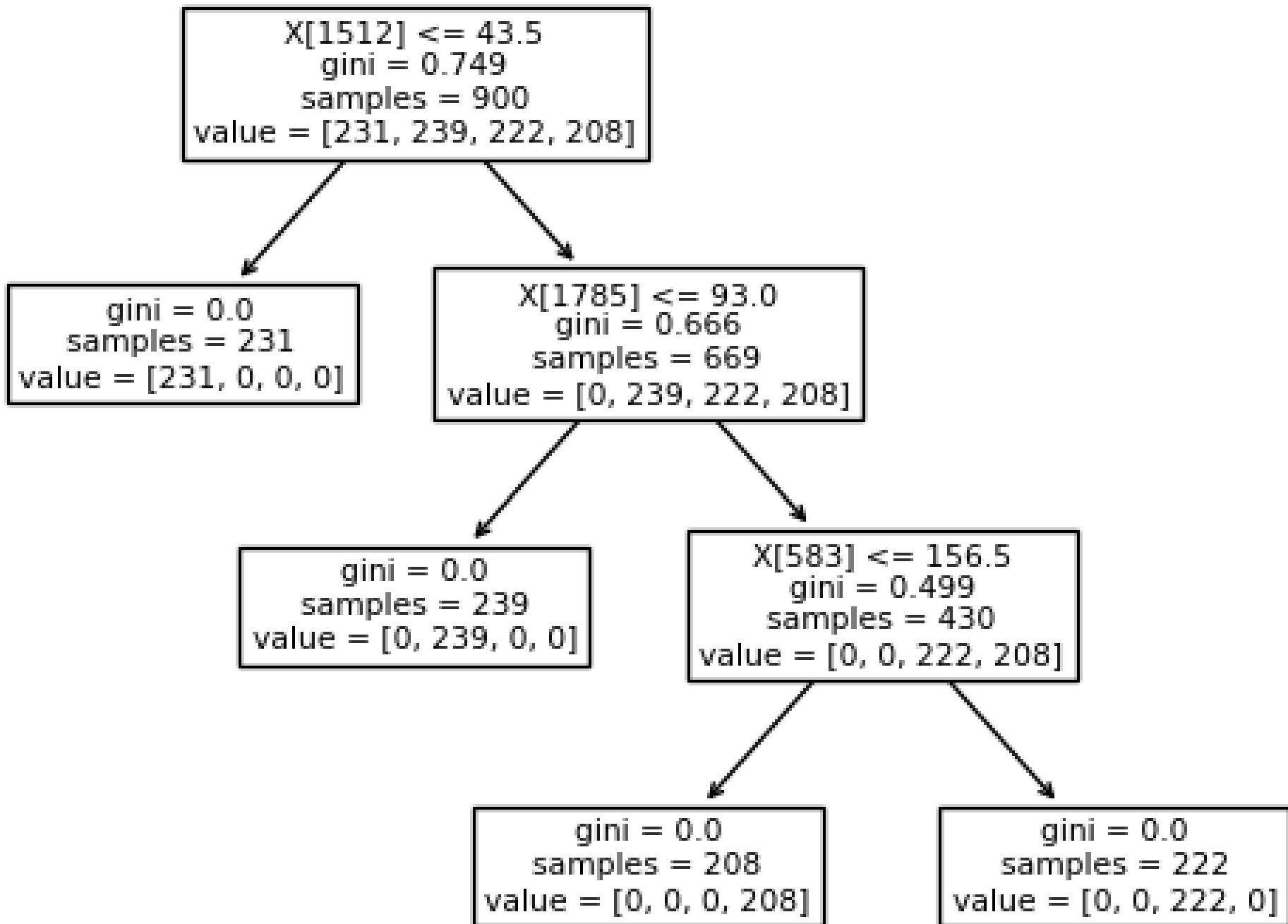
- Fast convergence
- Almost 100% accuracy

```
step 5, test accuracy 0.24
step 10, training accuracy 0.28
step 10, test accuracy 0.3
step 15, training accuracy 0.38
step 15, test accuracy 0.44
step 20, training accuracy 0.76
step 20, test accuracy 0.8
step 25, training accuracy 0.96
step 25, test accuracy 0.96
step 30, training accuracy 1
step 30, test accuracy 0.98
step 35, training accuracy 1
step 35, test accuracy 1
step 40, training accuracy 1
step 40, test accuracy 1
step 45, training accuracy 1
step 45, test accuracy 1
step 50, training accuracy 1
step 50, test accuracy 1
|
```

# Using Decision Tree

---

- *Decision Tree* is Machine Learning algorithms that can perform classification , regression , as well as multioutput tasks.
- Decision Tree uses a white box model - opposite to black box model algorithms it is easy to interpret its predictions
- Making predictions requires traversing DT from the root to the leaf : so the algorithm compares all features on all samples at each node , (or less, if max\_feature is not set )



```
/Users/maleykaseyidova/opt/anaconda3/envs/tf37/bin  
— 0.003290891647338867 seconds —  
Test Acc: 1.0
```

Test accuracy = 100%

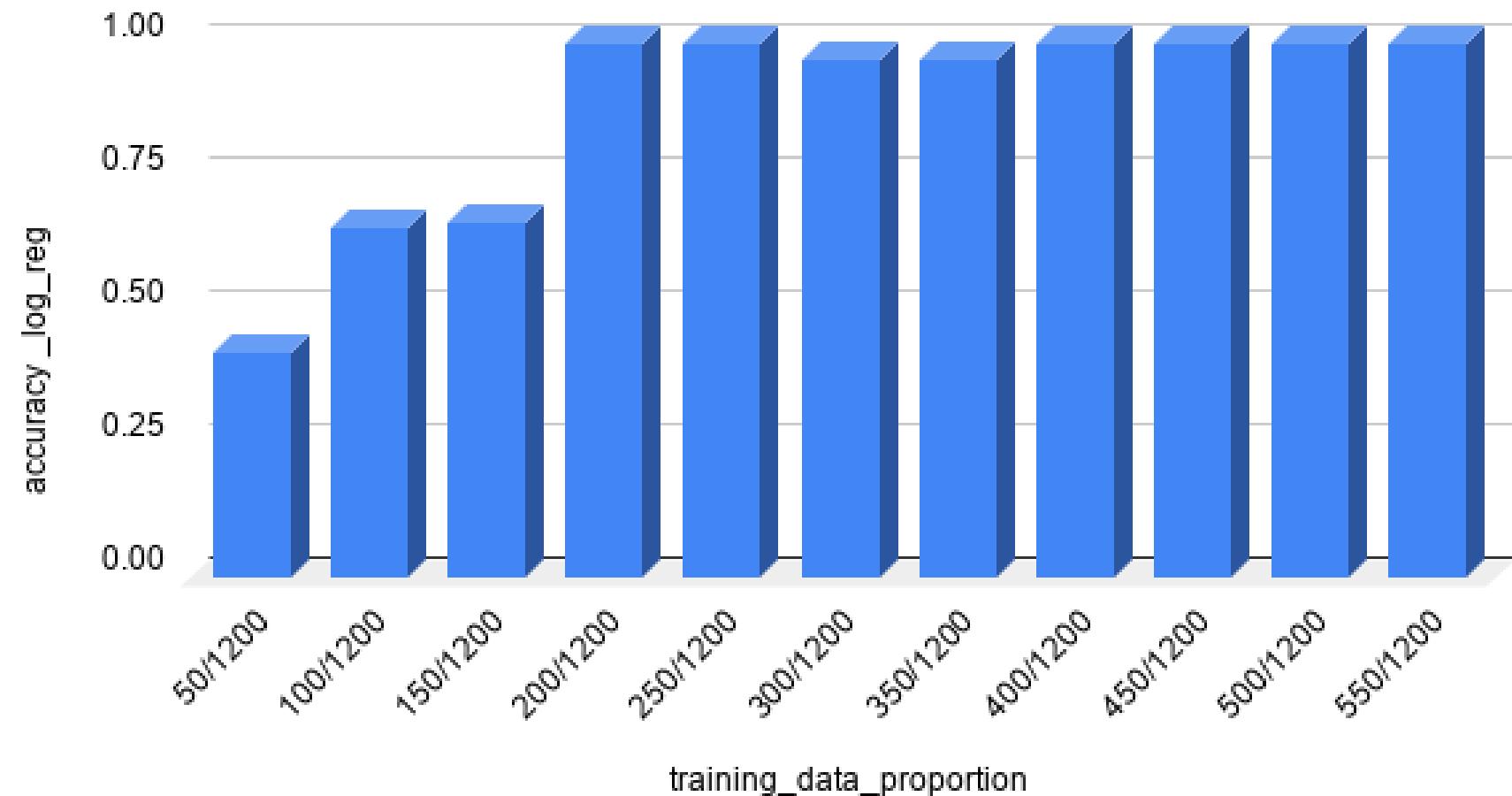
Execution time for 300 test samples: 0.003 seconds

---

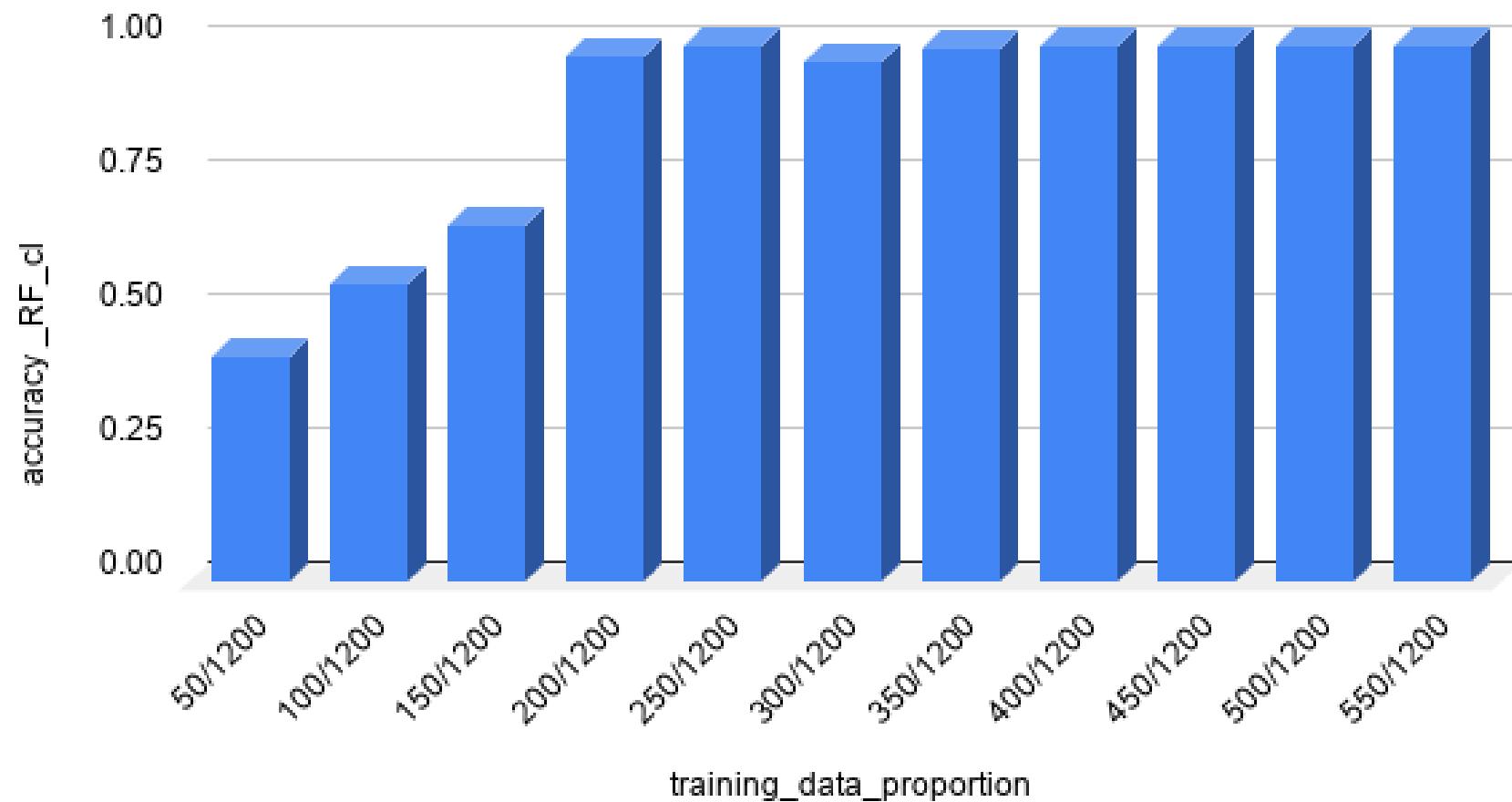
## Using Voting Classifier :

training_data_proportion	acc_log_reg_cl	acc_RF_cl	acc_SCV_cl	acc_V_cl
50/1200	0.4166666667	0.4166666667	0.4166666667	0.4166666667
100/1200	0.652173913	0.5543478261	0.652173913	0.652173913
150/1200	0.6666666667	0.6666666667	0.6666666667	0.6666666667
200/1200	1	0.9820359281	0.9520958084	0.994011976
250/1200	1	1	0.9848484848	1
300/1200	0.9688888889	0.9688888889	0.9511111111	0.9688888889
350/1200	0.9677419355	0.9959677419	0.9838709677	0.9959677419
400/1200	1	1	0.9850187266	1
450/1200	1	1	0.9858156028	1
500/1200	1	1	0.9863013699	1
550/1200	1	1	0.9899328859	1

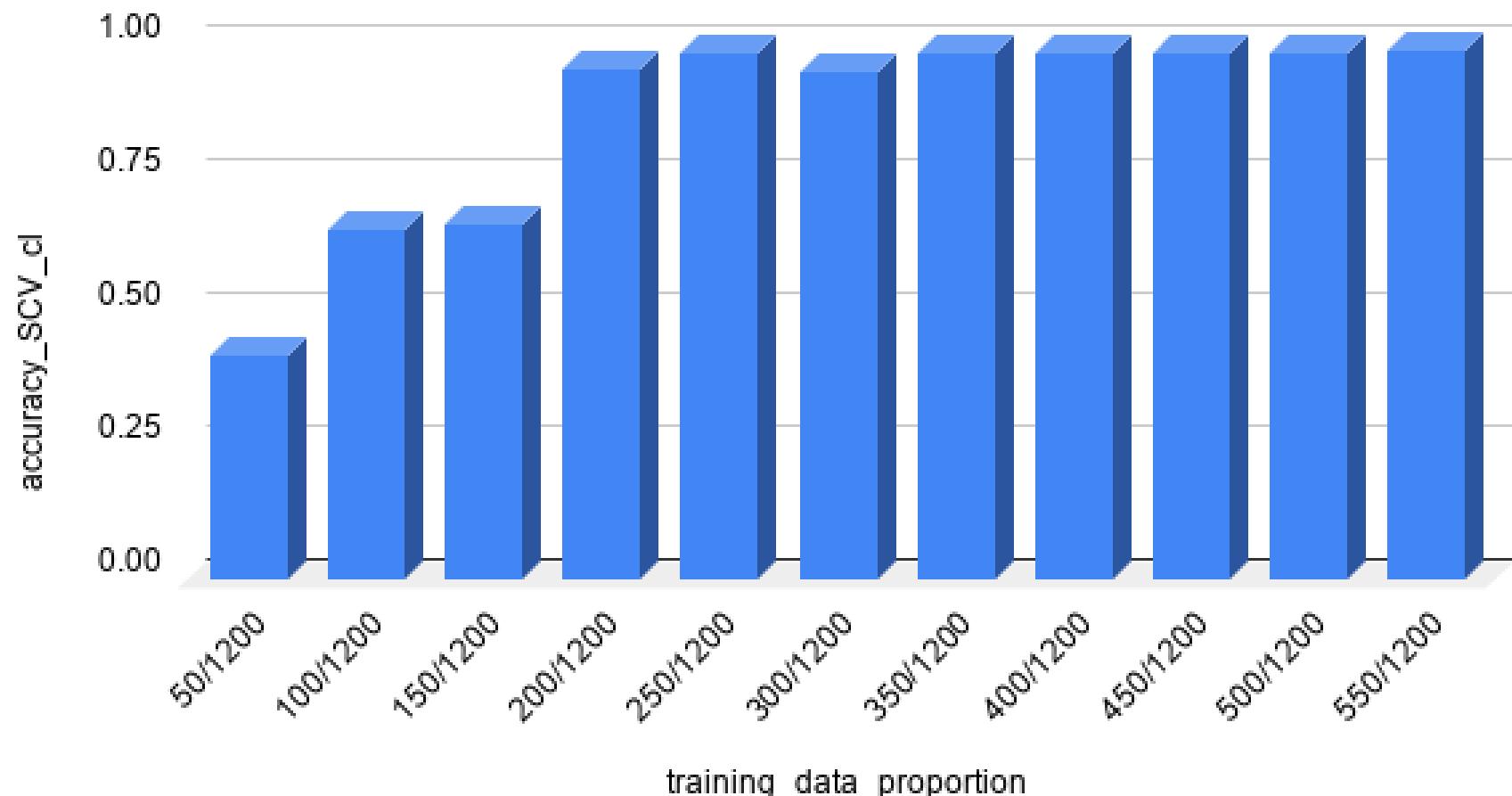
accuracy\_log\_reg vs training\_data\_proportion



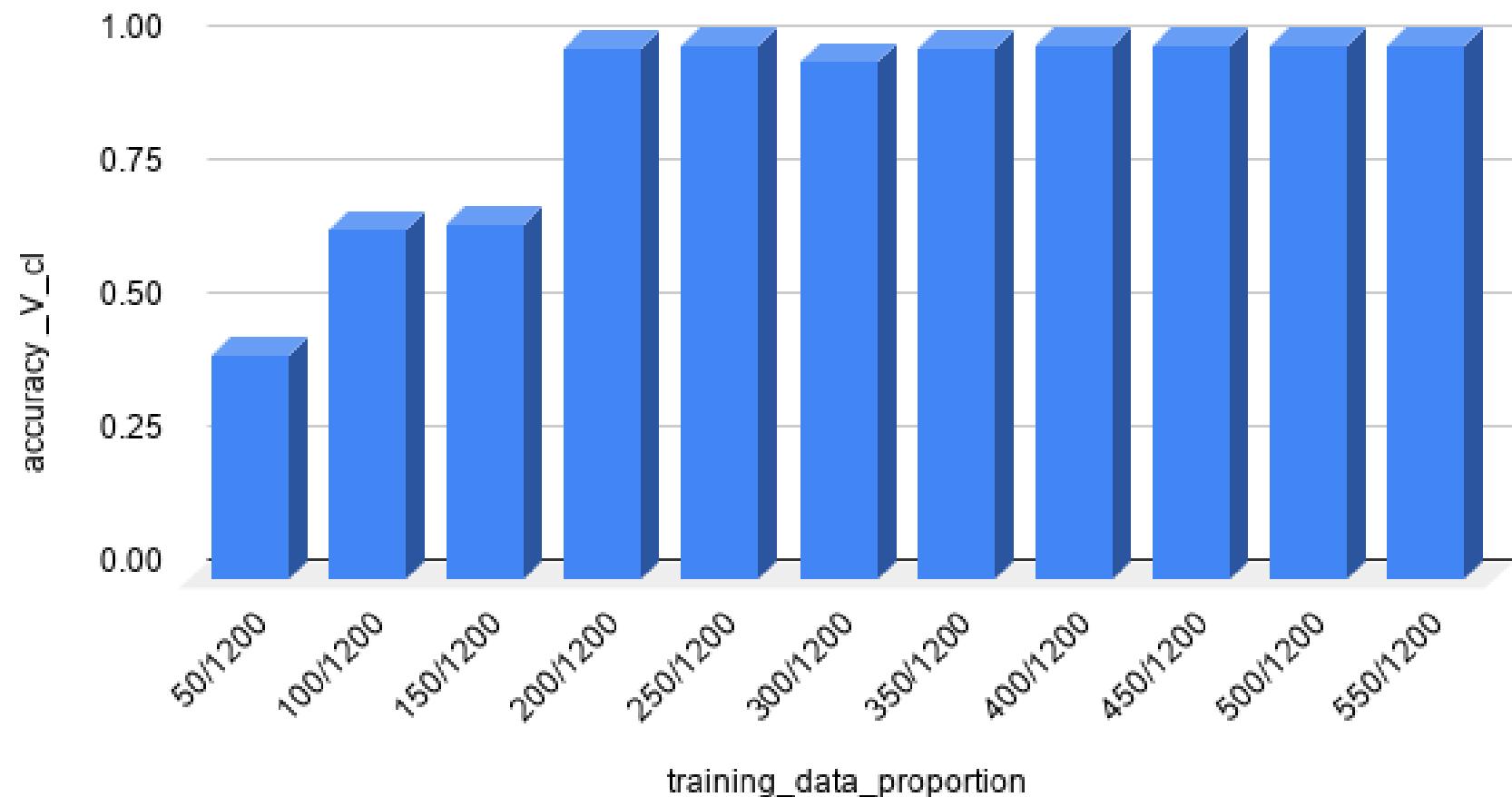
accuracy \_RF\_cl vs training\_data\_proportion



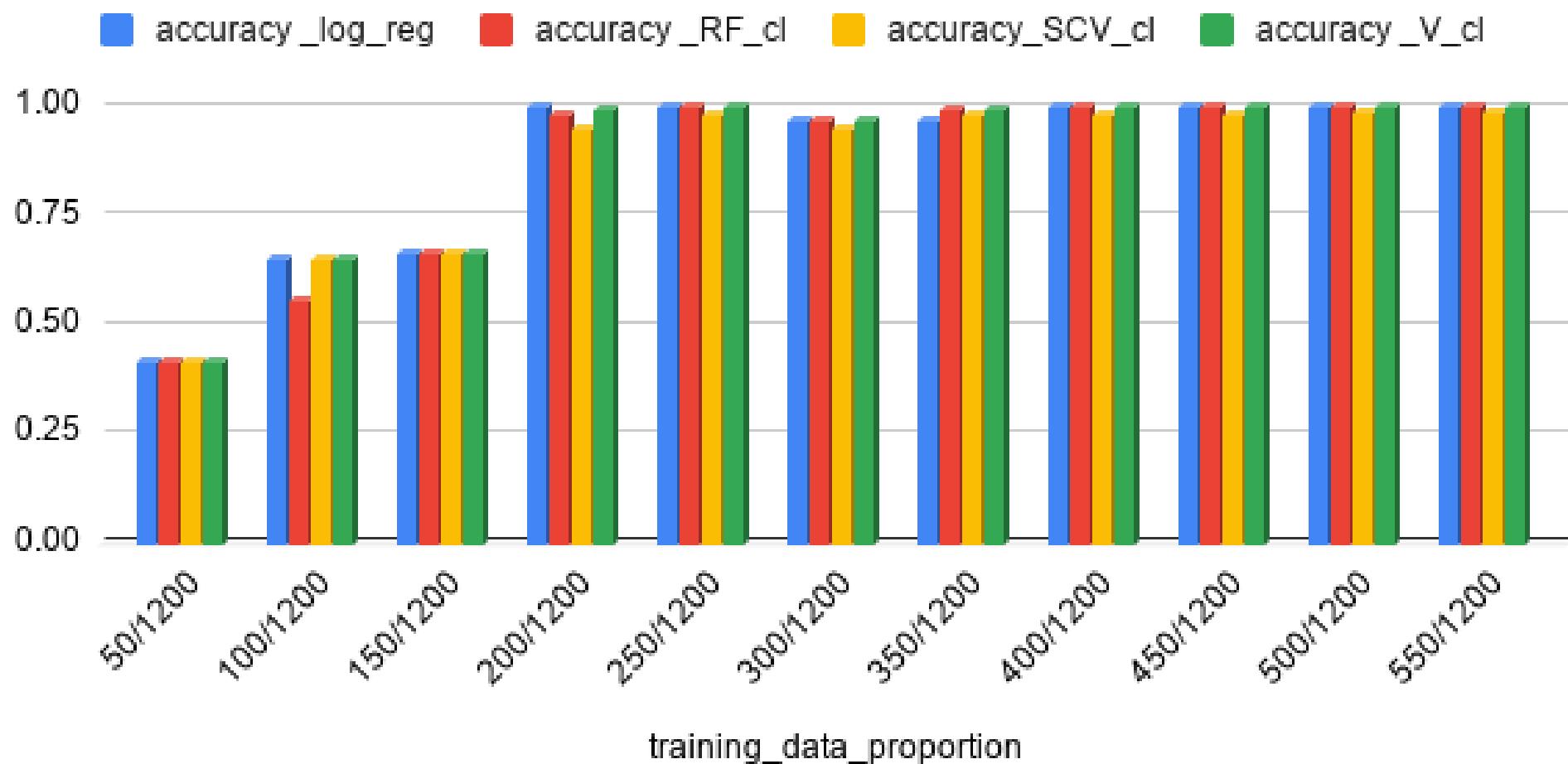
accuracy\_SCV\_cl vs training\_data\_proportion



accuracy \_V\_cl vs training\_data\_proportion



accuracy\_log\_reg, accuracy\_RF\_cl, accuracy\_SCV\_cl and  
accuracy\_V\_cl



*Grid search result :*

**max\_features : 2**  
**n\_estimators : 10**

```
param_grid=[{'max_features': [2, 4, 6, 8],  
            'n_estimators': [3, 10, 30]},  
            {'bootstrap': [False], 'max_features': [2, 3, 4],  
             'n_estimators': [3, 10]}],  
            pre_dispatch='2*n_jobs', refit=True, return_train_score=True,  
            scoring='neg_mean_squared_error', verbose=0)  
best parameters: {'max_features': 2, 'n_estimators': 10}
```