(1)

$$k(x, x') = \exp(-\|x - x'\|^2 / 2\gamma^2)$$

$$= \exp\left(-\frac{(x - x')^2}{2\gamma^2}\right)$$

$$= e^{-\frac{x^2 - x'^2}{2\gamma^2}} \qquad \left[1 + \frac{2xx'}{1!(2\gamma^2)} + \frac{(2xx')^2}{2!(4\gamma^2)} \cdots\right)$$

$$= \phi(x_i)^T \phi(x')$$

Amuya Hampihallikar
CMPSC 448
HW 4

(2) a) $y_i(w^T x_i + b) \geq 1 - \xi_{pi}$ dual variables $\alpha_1 \dots \alpha_n$
$\xi_i \geq 0$ dual variables $\beta_1 \dots \beta_n$.

$$\min \|w\|_2^2 + C \hat{\sum}_1 \xi_i$$

Lagragian function

$$d(x, \alpha) \quad f(x) + \hat{\sum}_1 \alpha_i g_i(x)$$

$$d_1 = \|w\|_2^2 + C \hat{\sum}_1^n \xi_i + \hat{\sum}_1 \alpha_i (1 - \xi_i - y(w^T x + b))$$

$$d_2 = \|w\|_2^2 + C \hat{\sum}_1 \xi_{pi} + \hat{\sum}_1 \alpha_i (1 - \xi_i - y(w^T x + b)) + \hat{\sum}_1^n \beta_i (-\xi_{pi})$$

b) KKT optimality conditions

$$\nabla f(x^*) + \Sigma \alpha_i^* \nabla g_i(x^*) = 0$$

$$\nabla d = 0$$

$$\Rightarrow \frac{\partial L}{\partial \alpha} = 0 \quad \frac{\partial L}{\partial \beta} = 0$$

$$\Sigma \alpha^* (1 - \xi_{pi}^* - y_i (w_i^T x_i^* + b)) = 0$$

$$\Sigma \beta_i^* (-\xi_i^*) = 0$$

c) $\frac{\partial L}{\partial w} = 0$

$\Rightarrow w_i - \sum_1^n \alpha_i y_i x_i = 0$

$\Rightarrow w_i = \sum_1^n \alpha_i y_i x_i$

$\frac{\partial L}{\partial \xi_i} = 0$

$\Rightarrow -\alpha_i - \beta_i = 0$

$\alpha_i = -\beta_i$

$\frac{\partial L}{\partial b} = 0$

$\Rightarrow \sum_1^n \alpha_i y_i = 0$

$\frac{\partial L}{\partial \alpha_i} = 0$

$\Rightarrow \sum_1^n [1 - \xi_i - y_i(w^T x_i + b) + \xi_i] = 0$

$\Rightarrow$

for $0 \leq \alpha_i \leq C$.

from kernal trick

$$\max_{\alpha \in R^n} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_1^n \alpha_i \alpha_j y_i y_j \beta_i \beta_j$$

$$\text{s.t. } \sum_1^n \alpha_i y_i = 0$$

$$\alpha_i \in [0, C] \quad i = 1, 2, \ldots n.$$

(3) a)

```
if (Color = Yellow):
    predict Eadible = Yes.
else:
    predict Eadible = No.
```

decision stump f_1

in Ensemble classifier

$f(x) = \sum_1^K \alpha_i f_i(x)$

initial weights = [1/16, 1/16, 1/16, 1/16, 1/16, 1/16
1/16, 1/16, 1/16, 1/16, 1/16, 1/16
1/16, 1/16, 1/16, 1/16]

from table we know f_1 will give incorrect predictions for D_2, D_3, D_{10}, D_{12}, D_{13}, D_{14}

$\epsilon_1 = \sum_1^K w_i^0$ such that $f_1(x_i) \neq y_i$

$\epsilon_1 = 6/16$

$\alpha = \frac{1}{2} \log\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = \frac{1}{2} \log\left(\frac{10/16}{6/16}\right)$

$= \frac{1}{2} \log 5/3$

b) Next step in AdaBoost is recomputing data weights depending on error of f_1

Weights in second instance of iteration

Update rule: $\frac{1}{Z} w_i^{(0)} \exp(-\alpha_1 y_i f_1(x_i))$

$Z = 6\sqrt{10/6} + 10\sqrt{6/10}$

$= 6\sqrt{5/3} + 10\sqrt{6/10}$

$\therefore$ for correct prediction $\sqrt{3/5}/Z$

$\therefore$ for incorrect prediction $\sqrt{5/3}/Z$

c) We shouldn't stop the iteration if the error rate of the current weak classifier on weighted training data as this will result in overfitting on the training data. We will require a validation set to prevent overfitting.

Ameya Hampihallikar

CMPSC 448

HW4

Question 4

Part 1

The two classifiers I chose were Boosted Decision Trees and Random Forests

Boosted Decision trees

Boosting refers to a general and provably effective method of producing a very accurate classifier by combining rough and moderately inaccurate rules of thumb. The main objective is to learn a single (weak) classifier, learn many weak classifiers that are good at different parts of the input space.

Algorithm

On each iteration k = 1, 2, …, K (number of weak learners)

a. Weight each training examples by how incorrectly it was classified (i.e, hardness of classifying example)

b. Learn a week classifier on weighted training data

c. Compute the strength of this classifier alpha(k)

Return final classifier f(x) = sign(sigma(alpha(k)f(k)(x))

Random Forests

Random forest will contain a large number of individual decision trees; in turn they act as an ensemble. This helps to obtain better predictive performance.

Algorithm

a. Random forests provide an improvement over bagged trees by way of a small tweak that de-correlates the models.

b. **B**uild a number of decision trees on bootstrapped training samples.

c. While building these decision trees, each time a split in a tree is considered, a random sample of features is chosen as split candidates from the full set of all features.

d. But when building these decision trees, each time a split in a tree is considered, a random sample of features (predictors) is chosen as split candidates from the full set of all features.

e. Each time a split in a tree is considered, a random sample of p < d features (predictors) is chosen as split candidates from the full set of all d features.

e. The split is allowed to use only one of those p features. A fresh sample of features is taken at each split, and typically we choose p = sqrt(d).

Part 2

Training methodology for both programs:

1. load training and testing datasets

2. Choose different sets of values for each hyperparameter you choose to tune

3. Run GridSearch on the grid of the sets of hyperparameter values

4. get the best combination of values, cross_val_scores

    ->The GridSearchCV utility function does k fold cross validation implicitly.

5. Create your model based on the hyperparameters returned by GridSearch

6. Fit the training data to your model

7. Calculate your accuracy

    -> I have compared my results with the results from a model with default parameters

Code for Random Forest classfier

[1]
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score


from sklearn.ensemble import RandomForestClassifier
```

[2]
```
# load data in LibSVM sparse data format
training_data = load_svmlight_file
(r"C:\Users\ameya\Documents\projects\cmpsc448_work\HW4\a9a.txt")
X_train = training_data[0]
y_train = training_data[1]
```

[4]
```
testing_data = load_svmlight_file
(r"C:\Users\ameya\Documents\projects\cmpsc448_work\HW4\a9a.t")
X_test = testing_data[0]
y_test = testing_data[1]
```

[3]
```
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

[5]

```
n_estimators = [50, 100, 200, 300]
bootstrap = [True, False]
max_depth = [10 ,50, 100]
min_impurity_decrease = [0.0, 0.05, 0.1]
min_samples_leaf = [1, 5, 10]

param_grid = dict(n_estimators = n_estimators, bootstrap = bootstrap,
 max_depth = max_depth, min_impurity_decrease =
min_impurity_decrease, min_samples_leaf = min_samples_leaf)
```

[6]

```
grid = GridSearchCV(model, param_grid, refit=True, cv =3, verbose=2)
grid.fit(X_train, y_train)
```

```
les_leaf=10, n_estimators=100; total time=   0.4s
[CV] END bootstrap=False, max_depth=100, min_impurity_decrease=0.1, min_samp
les_leaf=10, n_estimators=200; total time=   0.9s
[CV] END bootstrap=False, max_depth=100, min_impurity_decrease=0.1, min_samp
les_leaf=10, n_estimators=200; total time=   0.9s
[CV] END bootstrap=False, max_depth=100, min_impurity_decrease=0.1, min_samp
les_leaf=10, n_estimators=200; total time=   0.9s
[CV] END bootstrap=False, max_depth=100, min_impurity_decrease=0.1, min_samp
les_leaf=10, n_estimators=300; total time=   1.4s
[CV] END bootstrap=False, max_depth=100, min_impurity_decrease=0.1, min_samp
les_leaf=10, n_estimators=300; total time=   1.3s
[CV] END bootstrap=False, max_depth=100, min_impurity_decrease=0.1, min_samp
les_leaf=10, n_estimators=300; total time=   1.4s
Output was trimmed for performance reasons.
To see the full output set the setting "jupyter.textOutputLimit" to 0.
...

GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid={'bootstrap': [True, False], 'max_depth': [10, 50, 1
00],
                         'min_impurity_decrease': [0.0, 0.05, 0.1],
                         'min_samples_leaf': [1, 5, 10],
                         'n_estimators': [50, 100, 200, 300]},
             verbose=2)
```

[7]

```
print(grid.best_params_)
```

```
{'bootstrap': False, 'max_depth': 100, 'min_impurity_decrease': 0.0, 'min_samp
les_leaf': 5, 'n_estimators': 100}
```

[8]

```
# model_best = RandomForestClassifier(bootstrap = False, max_depth =
50, min_impurity_decrease= 0.0, min_samples_leaf=100,
n_estimators=200)
model_best = RandomForestClassifier(bootstrap = True, max_depth = 20,
 min_impurity_decrease= 0.0, min_samples_leaf=1, n_estimators=100)
model_best.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=20)
```

[9]

```
# make predictions for test data
y_pred = model_best.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))

model_default = RandomForestClassifier()
model_default.fit(X_train, y_train)
# make predictions for test data
y_pred = model_default.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)


print("Default Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 84.52%
Default Accuracy: 83.31%
```

Code for XGBoost

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

from xgboost import XGBClassifier
```

[2]

```
# load data in LibSVM sparse data format
training_data = load_svmlight_file
(r"C:\Users\ameya\Documents\projects\cmpsc448_work\HW4\a9a.txt")
X_train = training_data[0]
y_train = training_data[1]
```

[3]

```
testing_data = load_svmlight_file
(r"C:\Users\ameya\Documents\projects\cmpsc448_work\HW4\a9a.t")
X_test = testing_data[0]
y_test = testing_data[1]
```

[4]

```
#  fit model on training data
# for simplicity we fit based on default values of hyperparameters
model = XGBClassifier()
model.fit(X_train, y_train)
```

```
C:\Users\ameya\AppData\Local\Programs\Python\Python39\lib\site-packages\xgboost\s
klearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is depreca
ted and will be removed in a future release. To remove this warning, do the follo
wing: 1) Pass option use_label_encoder=False when constructing XGBClassifier obje
ct; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...,
[num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[17:30:44] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/
src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric use
d with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
```

[5]

```
n_estimators = [100, 150, 200, 250, 300]
max_depth = [4,5,6,7,8]
lambdareg = [1]
learning_rate = [0.1, 0.15, 0.2, 0.25, 0.3]
missing = []
objective = ['binary:logistic', 'binary:logitraw', 'binary:hinge']

param_grid = dict(n_estimators = n_estimators, max_depth = max_depth,
lambdareg = lambdareg, learning_rate = learning_rate, objective =
objective)
```

[6]

```
grid = GridSearchCV(model, param_grid, refit=True, cv=3, verbose=2)
grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=3,
             estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                     colsample_bylevel=1, colsample_bynode=1,
                                     colsample_bytree=1, gamma=0, gpu_id=-1,
                                     importance_type='gain',
                                     interaction_constraints='',
                                     learning_rate=0.300000012,
                                     max_delta_step=0, max_depth=6,
                                     min_child_weight=1, missing=nan,
                                     monotone_constraints='()',
                                     n_estimators=100, n_jobs=8,
                                     num_parallel_tree=1, random_state=0,
                                     reg_alpha=0, reg_lambda=1,
                                     scale_pos_weight=1, subsample=1,
                                     tree_method='exact', validate_parameters=
1,
                                     verbosity=None),
             param_grid={'lambdareg': [1],
                         'learning_rate': [0.1, 0.15, 0.2, 0.25, 0.3],
                         'max_depth': [4, 5, 6, 7, 8],
                         'n_estimators': [100, 150, 200, 250, 300],
                         'objective': ['binary:logistic', 'binary:logitraw',
                                       'binary:hinge']},
             verbose=2)
```

[7]

```
print(grid.best_params_)
```

```
{'lambdareg': 1, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 300, 'obj
ective': 'binary:logistic'}
```

[9]

```
model_best = XGBClassifier(learning_rate=0.1, max_depth=4,
n_estimators=300, objective='binary:logistic', reg_lambda=1)

model_best.fit(X_train, y_train)
```

```
C:\Users\ameya\AppData\Local\Programs\Python\Python39\lib\site-packages\xgboost
\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is depr
ecated and will be removed in a future release. To remove this warning, do the f
ollowing: 1) Pass option use_label_encoder=False when constructing XGBClassifier
object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2,
..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[18:34:23] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'logloss'.
Explicitly set eval_metric if you'd like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.1, max_delta_step=0, max_depth=4,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=300, n_jobs=8, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

[11]

```
# make predictions for test data
y_pred = model_best.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))



model_default = XGBClassifier()
model_default.fit(X_train, y_train)
# make predictions for test data
y_pred = model_default.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy = accuracy_score(y_test, predictions)
print("Default Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 85.20%
[18:35:34] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/
src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric use
d with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Exp
licitly set eval_metric if you'd like to restore the old behavior.
C:\Users\ameya\AppData\Local\Programs\Python\Python39\lib\site-packages\xgboost\s
klearn.py:1146: UserWarning: The use of label encoder in XGBClassifier is depreca
ted and will be removed in a future release. To remove this warning, do the follo
wing: 1) Pass option use_label_encoder=False when constructing XGBClassifier obje
ct; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ...,
[num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
Default Accuracy: 84.83%
```

[12]

```
crossVal = cross_val_score(model_best,X_train,y_train,cv=5).mean()

y_pred_train = model_best.predict(X_train)
predictions_train = [round(value) for value in y_pred_train]
train_acc = accuracy_score(y_train, predictions_train)

print("Cross_val_score: %.2f%%" % (crossVal * 100.0))
print("Training accuracy: %.2f%%" % (train_acc * 100.0))
```

```
ost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier
is deprecated and will be removed in a future release. To remove this warnin
g, do the following: 1) Pass option use_label_encoder=False when constructin
g XGBClassifier object; and 2) Encode your labels (y) as integers starting w
ith 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[22:37:00] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
C:\Users\ameya\AppData\Local\Programs\Python\Python39\lib\site-packages\xgbo
ost\sklearn.py:1146: UserWarning: The use of label encoder in XGBClassifier
is deprecated and will be removed in a future release. To remove this warnin
g, do the following: 1) Pass option use_label_encoder=False when constructin
g XGBClassifier object; and 2) Encode your labels (y) as integers starting w
ith 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[22:37:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Cross_val_score: 84.98%
Training accuracy: 86.15%
```

Part 3

Random Forest

From the scikit learn documentation

1. n_estimators
default=100
final value = 100
The number of trees in the forest.

2. Bootstrap
Default = True
final value = False
Choose whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

3. Max_depth
default=None
final value = 100
The maximum depth of the tree.

4. Min_impurity_decrease
Default = 0
final value = 0
A node will be split if this split induces a decrease of the impurity greater than or equal to this value.
The eqautipn is as follows
N_t / N * (impurity - N_t_R / N_t * right_impurity
- N_t_L / N_t * left_impurity)

5. Min_samples_leaf
default = 1
final value = 5
The minimum number of samples required to be at a leaf node.

XGBoost

From the XGBoost learn documentation

1. n_estimators
default=100
final value = 300
The number of trees in the forest.

2. max_depth
Default = 6
Final value = 4
Maximum depth of a tree

3. lambda
Default = 1
Final value = 1
L2 regularization term on weights. Increasing this value will make model more conservative.

4. learning_rate
Default = 0.3
Final value = 0.1
Step size shrinkage used in update to prevents overfitting

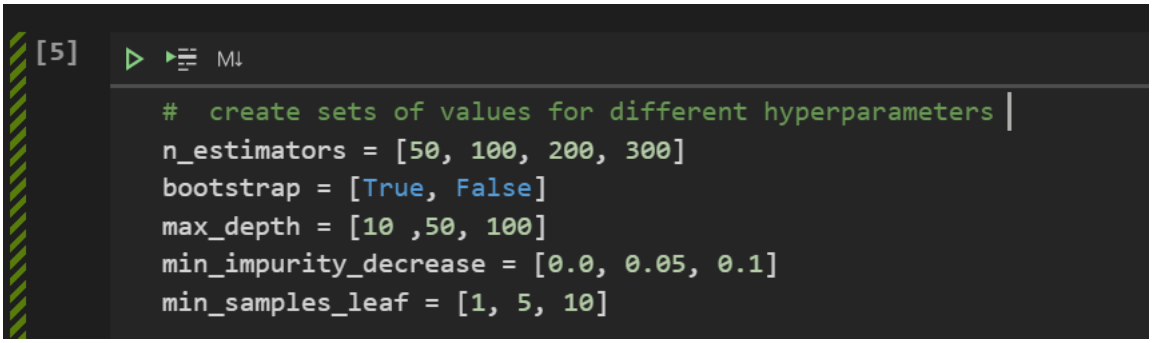
5. missing
Default = None
Final value = None
Handle the missing values.

6. objective
Default = reg: squared error
Final value = Binary: logistic
Objective function

Part 4

The settings used for the best models were the parameter configuration achieved form GridSearchCV.

Random Forests
1. Training accuracy = 91.68%
2. Cross-validation accuracy= 84.32%
3. Test accuracy = 84.52%

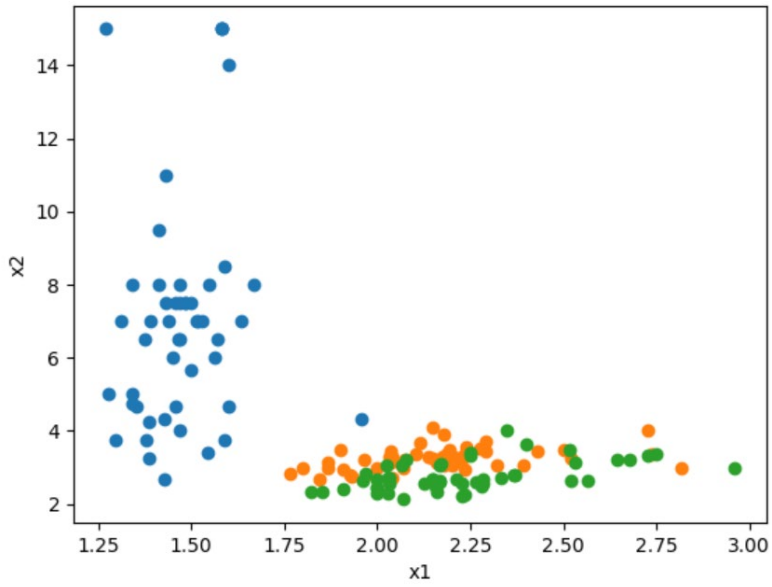


XGBoost

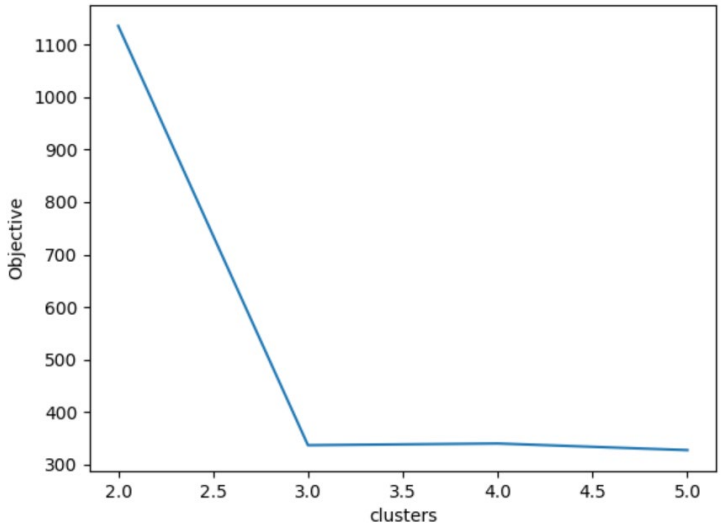
1. Training accuracy = 86.15%
2. Cross-validation accuracy= 84.98%
3. Test accuracy = 85.20%

```
n_estimators = [100, 150, 200, 250, 300]
max_depth = [4,5,6,7,8]
lambdareg = [1]
learning_rate = [0.1, 0.15, 0.2, 0.25, 0.3]
missing = []
objective = ['binary:logistic', 'binary:logitraw', 'binary:hinge']
```

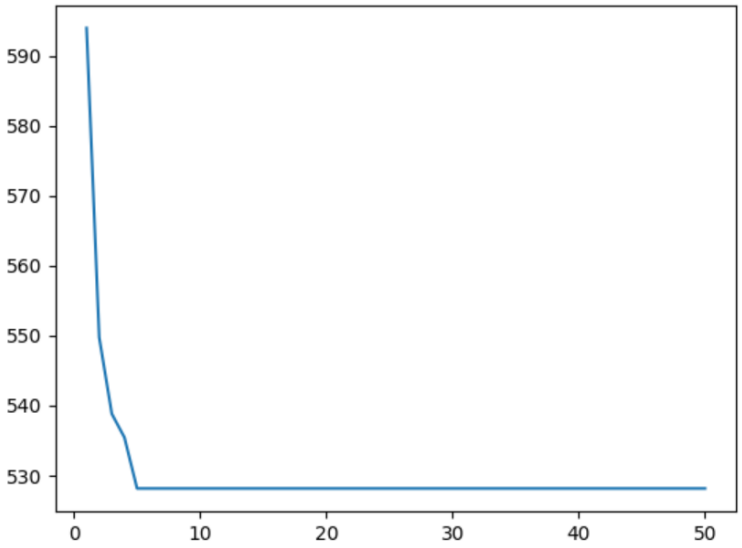
Question 5



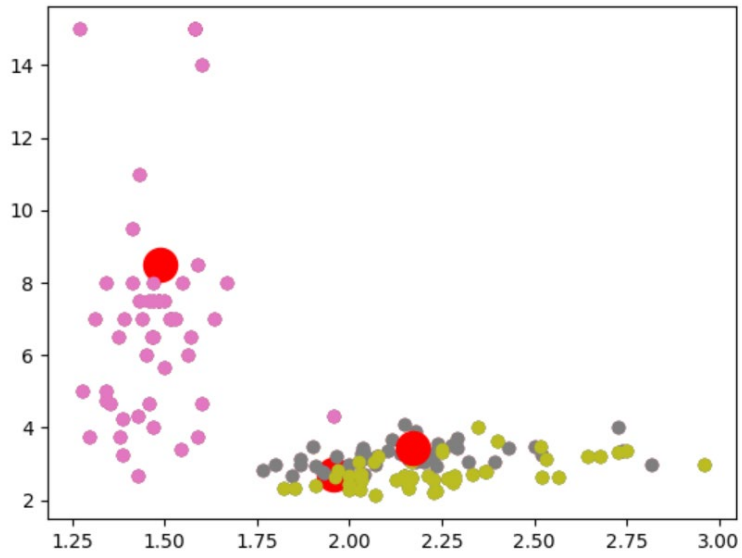
Plot of x1 vs x2



Plot of number of clusters vs clustering objective



Plot of objective changes vs. number of iterations



Data colored by assignment and cluster centers