1: The First Problem

(a) Algorithm:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
from sklearn.tree import DecisionTreeClassifier

car = pd.read_csv('car.csv', header=None)
car.columns = ['buying','maint','doors','persons','lug_boot','safety','evaluation']
car.describe()

x = car.iloc[:, :-1]
y = car.iloc[:, 6]
x.columns = ['buying','maint','doors','persons','lug_boot','safety']
y.columns = ['evaluation']
x.head()

x = pd.get_dummies(x, prefix_sep='_')
x = x.values
y = y.values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_st

# Logistic regression
clf = LogisticRegression(random_state=0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))

# Linear SVC
clf = SVC(kernel='linear', random_state=0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
print(classification_report(y_test,y_pred))

print('==========optimize==========\n')
clf = SVC(C=1.21, kernel='linear', random_state=0, tol=0.008)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))

# Rbf SVC
clf = SVC(kernel = 'rbf', random_state = 0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))

print('==========optimize==========\n')

clf = SVC(C=1.21, kernel = 'rbf', random_state = 0, tol=0.99)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))

# DT
clf = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

(b) Output:

In [89]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score,
from sklearn.tree import DecisionTreeClassifier
```

In [21]:

```
car = pd.read_csv('car.csv', header=None)
car.columns = ['buying','maint','doors','persons','lug_boot','safety','evaluation']
car.describe()
```

Out[21]:

	buying	maint	doors	persons	lug_boot	safety	evaluation
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	high	high	5more	4	small	high	unacc
freq	432	432	432	576	576	576	1210

In [43]:

```
x = car.iloc[:, :-1]
y = car.iloc[:, 6]
x.columns = ['buying','maint','doors','persons','lug_boot','safety']
y.columns = ['evaluation']
x.head()
```

Out[43]:

	buying	maint	doors	persons	lug_boot	safety
0	vhigh	vhigh	2	2	small	low
1	vhigh	vhigh	2	2	small	med
2	vhigh	vhigh	2	2	small	high
3	vhigh	vhigh	2	2	med	low
4	vhigh	vhigh	2	2	med	med

In [44]:

```
x = pd.get_dummies(x, prefix_sep='_')
x = x.values
y = y.values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_stat
```

In [159]:

```
# Logistic regression
clf = LogisticRegression(random_state=0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Acc:  0.9218523878437048
Testing Acc:  0.8988439306358381
[[ 67   2   9   1]
 [  6   7   0   4]
 [ 11   0 229   0]
 [  2   0   0   8]]
              precision    recall  f1-score   support

         acc       0.78      0.85      0.81        79
        good       0.78      0.41      0.54        17
       unacc       0.96      0.95      0.96       240
       vgood       0.62      0.80      0.70        10

    accuracy                           0.90       346
   macro avg       0.78      0.75      0.75       346
weighted avg       0.90      0.90      0.90       346
```

In [140]:

```
# Linear SVC
clf = SVC(kernel='linear', random_state=0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))

print('==========optimize==========\n')
clf = SVC(C=1.21, kernel='linear', random_state=0, tol=0.008)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Acc:  0.942836468885673
Testing Acc:  0.930635838150289
[[ 69   3   6   1]
 [  0  13   0   4]
 [ 10   0 230   0]
 [  0   0   0  10]]
              precision    recall  f1-score   support

         acc       0.87      0.87      0.87        79
        good       0.81      0.76      0.79        17
       unacc       0.97      0.96      0.97       240
       vgood       0.67      1.00      0.80        10

    accuracy                           0.93       346
   macro avg       0.83      0.90      0.86       346
weighted avg       0.93      0.93      0.93       346

==========optimize==========

Training Acc:  0.9442836468885673
Testing Acc:  0.9421965317919075
[[ 70   3   5   1]
 [  1  15   0   1]
 [  9   0 231   0]
 [  0   0   0  10]]
              precision    recall  f1-score   support

         acc       0.88      0.89      0.88        79
        good       0.83      0.88      0.86        17
       unacc       0.98      0.96      0.97       240
       vgood       0.83      1.00      0.91        10

    accuracy                           0.94       346
   macro avg       0.88      0.93      0.90       346
weighted avg       0.94      0.94      0.94       346
```

In [149]:

```
# Rbf SVC
clf = SVC(kernel = 'rbf', random_state = 0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))

print('==========optimize==========\n')

clf = SVC(C=1.21, kernel = 'rbf', random_state = 0, tol=0.99)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Acc:  0.9891461649782923
Testing Acc:  0.9508670520231214
[[ 74   4   0   1]
 [  0  13   0   4]
 [  7   0 233   0]
 [  1   0   0   9]]
              precision    recall  f1-score   support

         acc       0.90      0.94      0.92        79
        good       0.76      0.76      0.76        17
       unacc       1.00      0.97      0.99       240
       vgood       0.64      0.90      0.75        10

    accuracy                           0.95       346
   macro avg       0.83      0.89      0.85       346
weighted avg       0.96      0.95      0.95       346

==========optimize==========

Training Acc:  0.9949348769898697
Testing Acc:  0.9682080924855492
[[ 75   3   0   1]
 [  0  14   0   3]
 [  4   0 236   0]
 [  0   0   0  10]]
              precision    recall  f1-score   support

         acc       0.95      0.95      0.95        79
        good       0.82      0.82      0.82        17
       unacc       1.00      0.98      0.99       240
       vgood       0.71      1.00      0.83        10

    accuracy                           0.97       346
   macro avg       0.87      0.94      0.90       346
weighted avg       0.97      0.97      0.97       346
```

In [96]:

```
# DT
clf = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("Training Acc: ",clf.score(x_train, y_train))
print("Testing Acc: ", clf.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(classification_report(y_test,y_pred))
```

```
Training Acc:  1.0
Testing Acc:  0.9797687861271677
[[ 74   1   4   0]
 [  0  17   0   0]
 [  1   0 239   0]
 [  1   0   0   9]]
              precision    recall  f1-score   support

         acc       0.97      0.94      0.95        79
        good       0.94      1.00      0.97        17
       unacc       0.98      1.00      0.99       240
       vgood       1.00      0.90      0.95        10

    accuracy                           0.98       346
   macro avg       0.98      0.96      0.97       346
weighted avg       0.98      0.98      0.98       346
```