1.

```
import csv
from sklearn import linear_model
import matplotlib.pyplot as plt
import numpy as np

x = []
y = []

def get_data(filename):
    with open(filename,'r') as csvdocument:
        csvFileContent = csv.reader(csvdocument)
        next(csvFileContent)  # skipping column names
        for row in csvFileContent:
            x.append(int(row[0]))
            y.append(int(row[1]))
    return

get_data('datasets/pizzafranchise.csv')

model = linear_model.LinearRegression()
x = np.reshape(x,(len(x),1))
y = np.reshape(y,(len(y),1))
model.fit(x, y)

#predictions
predicted_cost = model.predict(900)
coeff = model.coef_[0][0]
const = model.intercept_[0]

#plotting
plt.scatter(x,y,color="yellow",label="Data Points")
plt.plot(x,model.predict(x),color="blue",label="Regression Line")
plt.scatter(900,model.predict(900),color="red",marker = "x",s=150,label="Predicted
Value")
plt.xlabel('annual franchise fee ($1000)')
plt.ylabel('start up cost ($1000)')
plt.legend()
plt.show()

print("The start up cost with franchise fee 900 is: $", str(predicted_cost[0][0]))
print("The regression coefficient is ", str(coeff), ", and the constant is ",
str(const))
print("the relationship equation between Franchise_Fee and Startup_Cost is:
Startup_Cost = ", str(coeff), "* Franchise_Fee + ", str(const))
```

2.

```
from collections import OrderedDict
import csv
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

i = []
s = []

def get_data(filename):
    with open(filename,'r') as csvdocument:
        csvFileContent = csv.reader(csvdocument)
        next(csvFileContent)  # skipping column names
        for row in csvFileContent:
            i.append(int(row[3]))
            s.append(int(row[4]))
    return
```

```
get_data('datasets/Customers.csv')
data = list(zip(i,s))
print("Data:")
print(data)

kmeans = KMeans(n_clusters=5)
kmeans.fit(data)

centroids = kmeans.cluster_centers_
labels = kmeans.labels_
print("Centroids:")
print(centroids)

colors = ["y","g","b","r","c"]
label = ["Cluster-1","Cluster-2","Cluster-3","Cluster-4","Cluster-5"]
#plot points
for i in range(len(data)):
    print("coordinate:",data[i], "label:", labels[i])
    plt.scatter(data[i][0], data[i][1], c = colors[labels[i]], label =
label[labels[i]])
#plot centroids
plt.scatter(centroids[:, 0],centroids[:, 1], label = "Centroids",marker = "x", s=150,
linewidths = 10, zorder = 15)
plt.title('clusters of customers')
plt.xlabel('Annual Income(k$)')
plt.ylabel('Spending Score(1-100)')
#remove duplicates of labels
handles, labels = plt.gca().get_legend_handles_labels()
by_label = OrderedDict(zip(labels, handles))
plt.legend(by_label.values(), by_label.keys())

plt.show()
#predictions
print("Predicted Class:")
print(kmeans.predict([(20,40)]))
```

3.

```
import numpy as np
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import metrics

diabetesdataset = datasets.load_wine()
x = diabetesdataset.data[:,:2]
y = diabetesdataset.target

h=0.3
xmin, xmax = x[:, 0].min() - 1, x[:, 0].max() + 1
ymin, ymax = x[:, 1].min() - 1, x[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(xmin, xmax, h),
                     np.arange(ymin, ymax, h))

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
model = svm.SVC()

predictions_linear = model.set_params(kernel='linear').fit(x_train,
y_train).predict(x_test)

predictions_rbf = model.set_params(kernel='rbf').fit(x_train, y_train).predict(x_test)

accuracy_linear = metrics.accuracy_score(y_test,predictions_linear)
```

```
accuracy_rbf = metrics.accuracy_score(y_test,predictions_rbf)

print("Accuracy with kernel=linear:",accuracy_linear)
print("Accuracy with kernel=rbf:",accuracy_rbf)
```

4.

```
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.tag import pos_tag
import nltk

# Read the file
f = open('datasets/input.txt',"r", encoding="UTF8")

#Tokenize words
sentence = f.read()
tokenized_words = word_tokenize(sentence)
print("Tokenized Words:\n",tokenized_words)

# Remove all the stop words
stop_words = stopwords.words('english')
filter_words = [w for w in tokenized_words if w not in stop_words]
filter_words = [w for w in filter_words if len(w)>2]
print("Filtered words after removing stop words\n",filter_words)

# Using Lemmatization
lemmatized_result = list()
for i in filter_words:
    lemmatized_result.append(WordNetLemmatizer().lemmatize(i))
print("Lemmatized Result\n",lemmatized_result)

# Using POS remove all the verbs
pos_result = list()
for i in pos_tag(lemmatized_result):
    if i[1][:2] == 'VB':
        continue
    else:
        pos_result.append(i[0])
print('POS output after removing verbs\n', pos_result)

# Calculate word frequency of the remaining
words_frequency = nltk.FreqDist(pos_result)
top_fivewords = dict()
for word, frequency in words_frequency.most_common(5):
    top_fivewords[word] = frequency

# Choose top five words
top_fivewords = top_fivewords.keys()
print('Top 5 words\n', top_fivewords)

# Find all the sentences with those most repeated words
output = list()
for lines in sentence.split('\n'):
    for word in top_fivewords:
        if word in lines.lower():
            output.append(lines)
            break

# Extract those sentences and concatenate
print('Summarization\n', "\n".join(output))
```