Bayesian Project Classification Code

December 17, 2018

0.1 Bayesian Statistics Project

0.1.1 Predicting Insurance Cost Using Bayesian Method

0.1.2 Classification Part

- Group Member: Mingzhong Gao(mgao24@jhu.edu), Chenyang Li(cli110@jhu.edu), Ziyan Lin(zlin25@jhu.edu), Han Wang(hwang178@jhu.edu), Weizhuo Wang (wwang111@jhu.edu)


```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

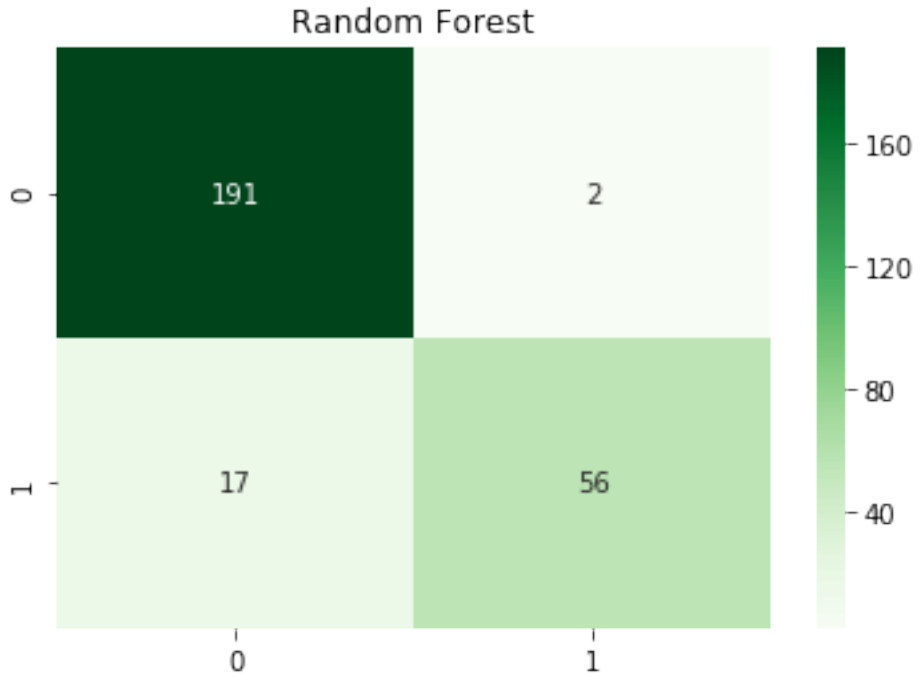
```
In [3]: trn = pd.read_csv('trn1.csv',header=0)
        tst = pd.read_csv('tst1.csv',header=0)
```

```
In [4]: x_train = trn[['V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V14','V
        y_train = trn['V1']
        x_test = tst[['V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V14','V1
        y_test = tst['V1']
        X = pd.DataFrame.append(x_train,x_test)
        Y = y_train.append(y_test)
```

```
In [5]: # Random Forest
        import warnings
        warnings.filterwarnings('ignore')
        random.seed(487)
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import f1_score,confusion_matrix
        import seaborn as sns
        from sklearn.model_selection import cross_val_score
        clf_rf = RandomForestClassifier()
        clr_rf = clf_rf.fit(x_train,y_train)
```

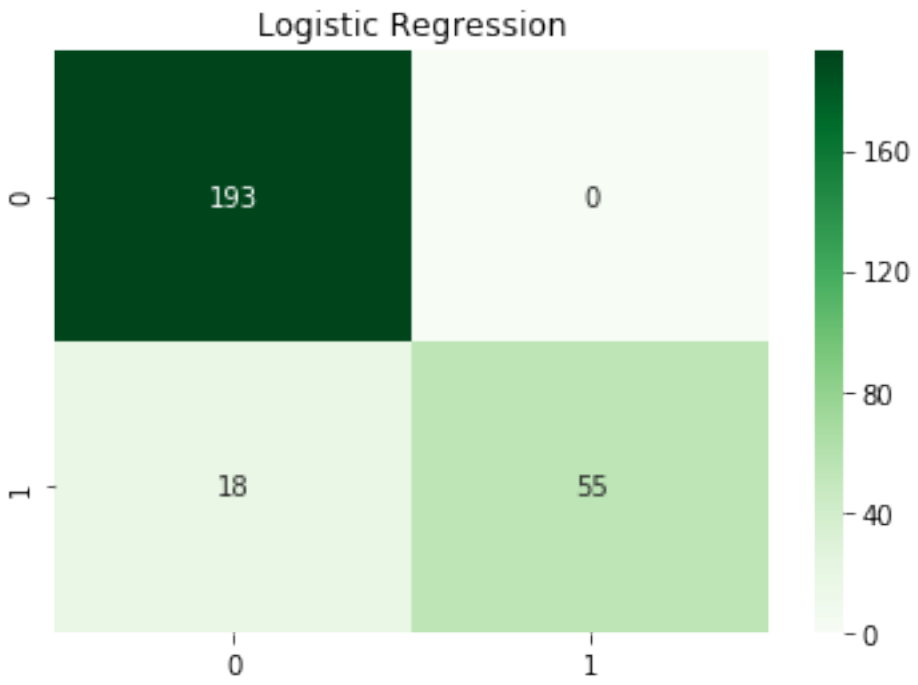
```
ac_rf = accuracy_score(y_test,clf_rf.predict(x_test))
cm_rf = confusion_matrix(y_test,clf_rf.predict(x_test))
f, ax = plt.subplots()
sns.heatmap(cm_rf,annot=True,fmt="d",cmap="Greens",ax=ax)
ax.set_title("Random Forest")
print('Accuracy is: ', np.mean(cross_val_score(clf_rf, X,Y, cv=10)))
```

```
Accuracy is:  0.9215351812366738
```



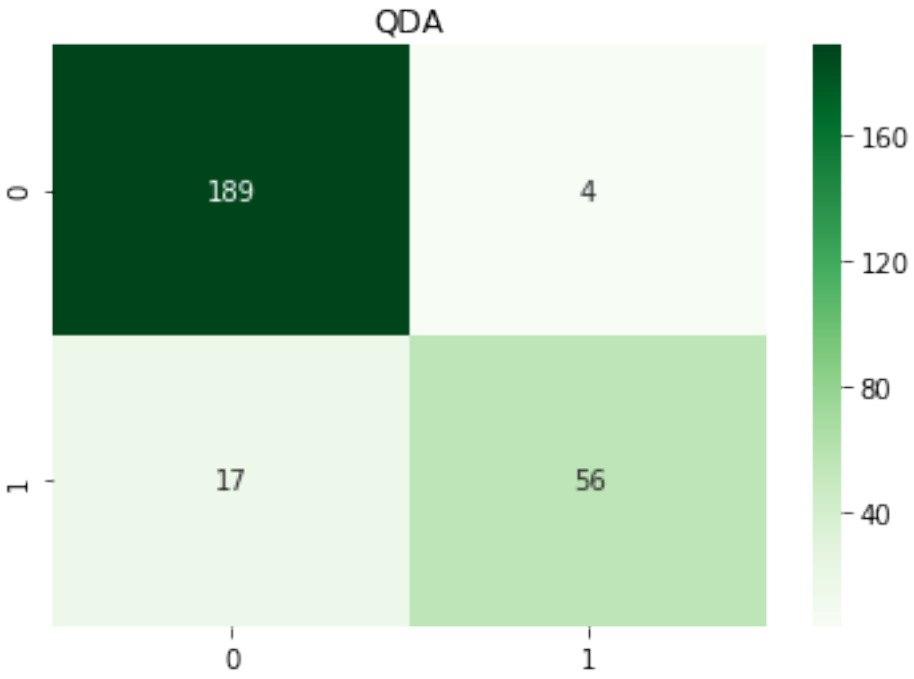
In [6]:
```
# Logistic Regression
import warnings
warnings.filterwarnings('ignore')
random.seed(487)
from sklearn.linear_model import LogisticRegression
clf_lr = LogisticRegression()
clr_lr = clf_lr.fit(x_train,y_train)
ac_lr = accuracy_score(y_test,clf_lr.predict(x_test))
cm_lr = confusion_matrix(y_test,clf_lr.predict(x_test))
f, ax = plt.subplots()
sns.heatmap(cm_lr,annot=True,fmt="d",cmap="Greens",ax=ax)
ax.set_title("Logistic Regression")
print('Accuracy is: ', np.mean(cross_val_score(clf_lr, X,Y, cv=10)))
```

```
Accuracy is:  0.9095892716866795
```



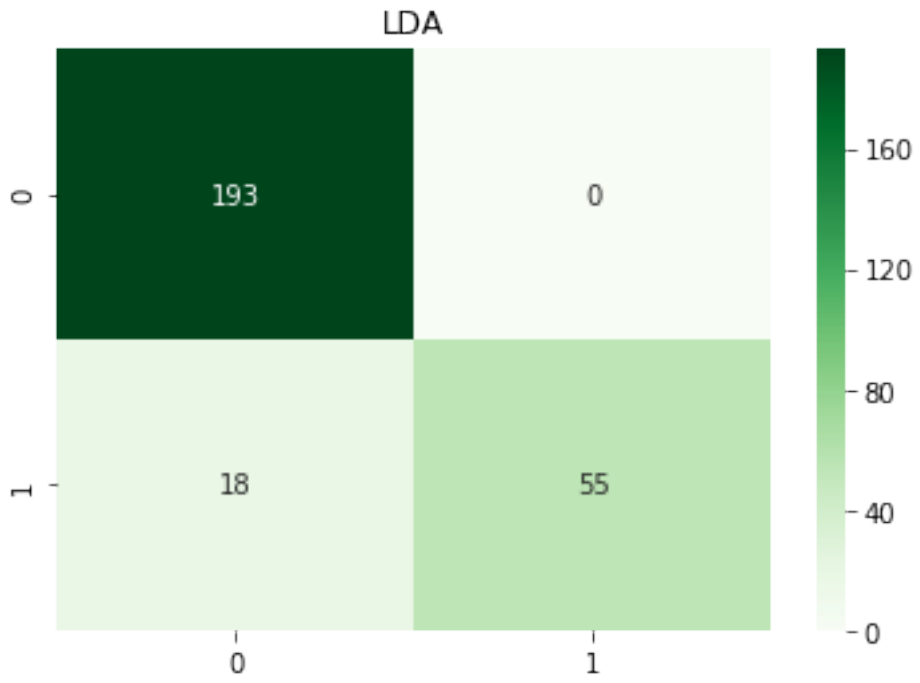
In [7]:
```
# Quadratic Discriminant Analysis
import warnings
warnings.filterwarnings('ignore')
random.seed(487)
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
clf_qda = QuadraticDiscriminantAnalysis()
clr_qda = clf_qda.fit(x_train,y_train)
ac_qda = accuracy_score(y_test,clf_qda.predict(x_test))
cm_qda = confusion_matrix(y_test,clf_qda.predict(x_test))
f, ax = plt.subplots()
sns.heatmap(cm_qda,annot=True,fmt="d",cmap="Greens",ax=ax)
ax.set_title("QDA")
print('Accuracy is: ', np.mean(cross_val_score(clf_qda, X,Y, cv=10)))
```

```
Accuracy is:  0.7528616316911682
```



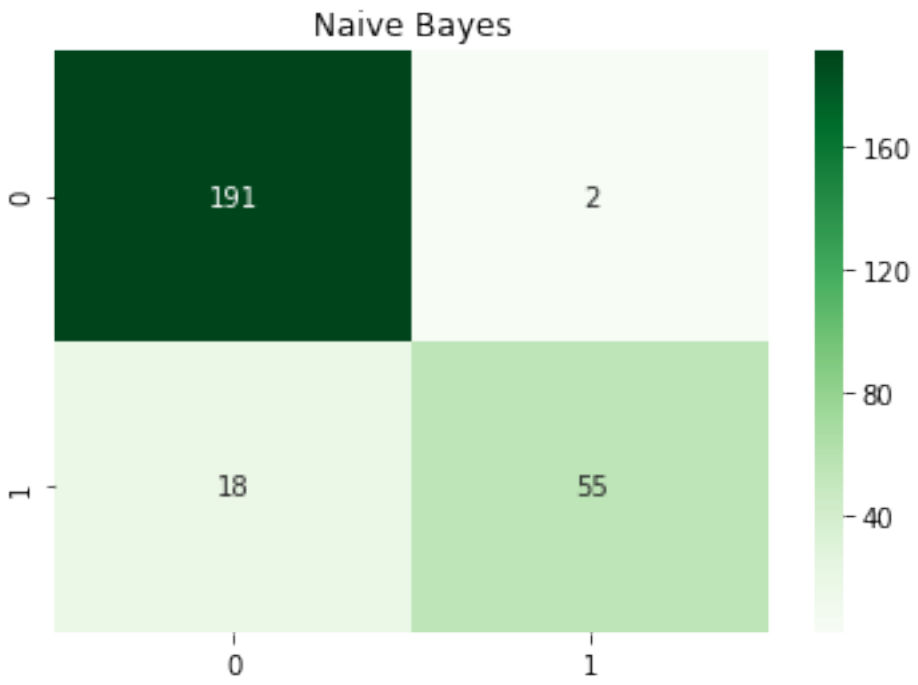
In [8]:
```
# Linear Discriminant Analysis
import warnings
warnings.filterwarnings('ignore')
random.seed(487)
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
clf_lda = LinearDiscriminantAnalysis()
clr_lda = clf_lda.fit(x_train,y_train)
ac_lda = accuracy_score(y_test,clf_lda.predict(x_test))
cm_lda = confusion_matrix(y_test,clf_lda.predict(x_test))
f, ax = plt.subplots()
sns.heatmap(cm_lda,annot=True,fmt="d",cmap="Greens",ax=ax)
ax.set_title("LDA")
print('Accuracy is: ', np.mean(cross_val_score(clf_lda, X,Y, cv=10)))
```

```
Accuracy is:  0.908843003029963
```



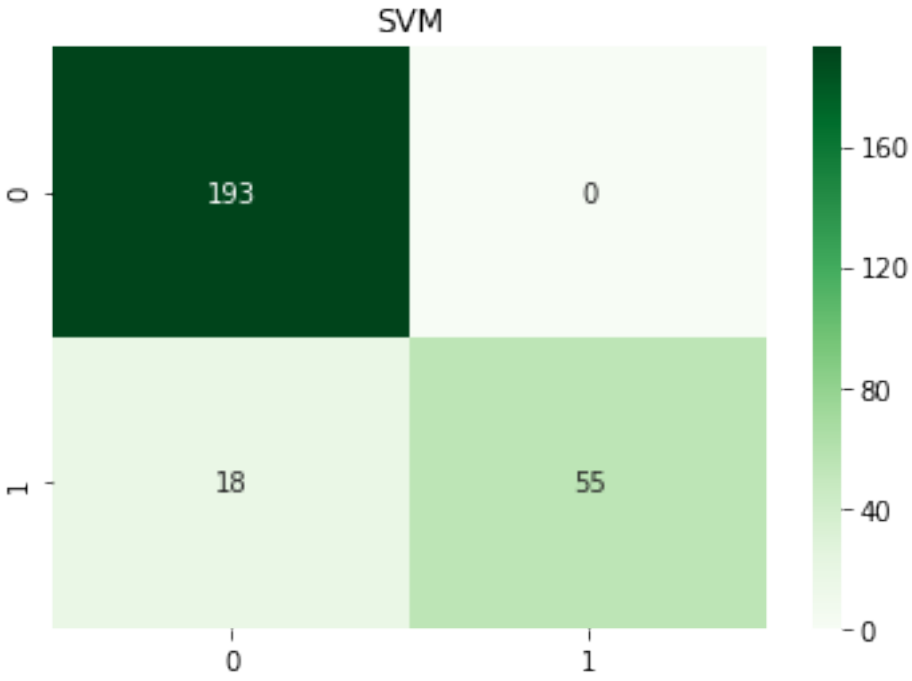

```
In [9]: # Naive Bayes
        import warnings
        warnings.filterwarnings('ignore')
        random.seed(487)
        from sklearn.naive_bayes import GaussianNB
        clf_nb = GaussianNB()
        clr_nb = clf_nb.fit(x_train,y_train)
        ac_nb = accuracy_score(y_test,clf_nb.predict(x_test))
        cm_nb = confusion_matrix(y_test,clf_nb.predict(x_test))
        f, ax = plt.subplots()
        sns.heatmap(cm_nb,annot=True,fmt="d",cmap="Greens",ax=ax)
        ax.set_title("Naive Bayes")
        print('Accuracy is: ', np.mean(cross_val_score(clf_nb, X,Y, cv=10)))
```

```
Accuracy is:  0.8991358994501178
```



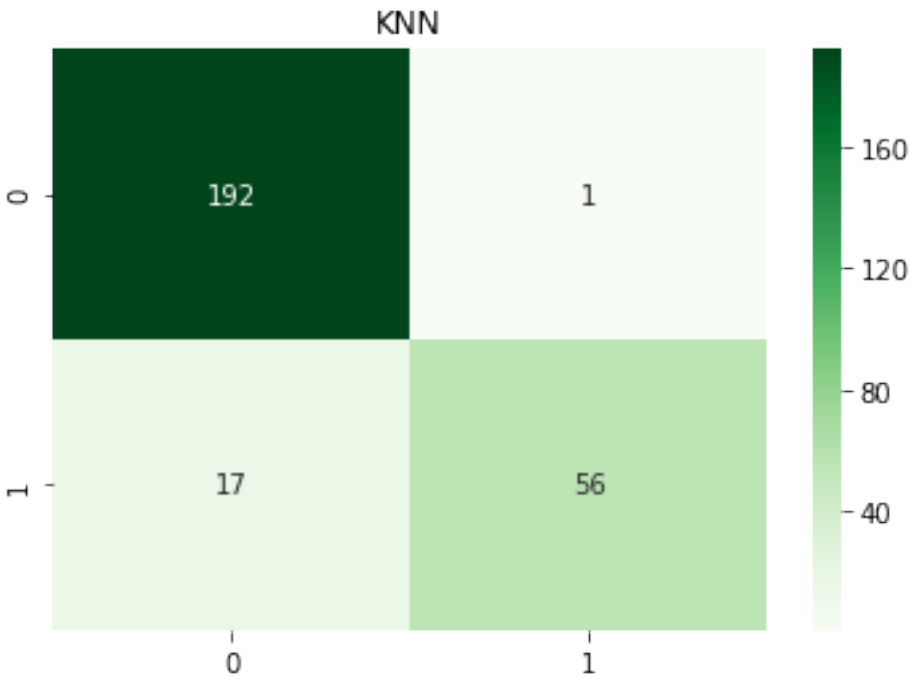

```
In [10]: # SVM
         import warnings
         warnings.filterwarnings('ignore')
         random.seed(487)
         from sklearn import svm
         clf_svm = svm.SVC(kernel='linear')
         clr_svm = clf_svm.fit(x_train,y_train)
         ac_svm = accuracy_score(y_test,clf_svm.predict(x_test))
         cm_svm = confusion_matrix(y_test,clf_svm.predict(x_test))
         f, ax = plt.subplots()
         sns.heatmap(cm_svm,annot=True,fmt="d",cmap="Greens",ax=ax)
         ax.set_title("SVM")
         print('Accuracy is: ', np.mean(cross_val_score(clf_svm, X,Y, cv=10)))
```

```
Accuracy is:  0.908843003029963
```



In [11]:
```
# KNN
import warnings
warnings.filterwarnings('ignore')
random.seed(487)
from sklearn import neighbors
clf_knn = neighbors.KNeighborsClassifier(5)
clr_knn = clf_knn.fit(x_train,y_train)
ac_knn = accuracy_score(y_test,clf_knn.predict(x_test))
cm_knn = confusion_matrix(y_test,clf_knn.predict(x_test))
f, ax = plt.subplots()
sns.heatmap(cm_knn,annot=True,fmt="d",cmap="Greens",ax=ax)
ax.set_title("KNN")
print('Accuracy is: ', np.mean(cross_val_score(clf_knn, X,Y, cv=10)))
```

```
Accuracy is:  0.9222702278083268
```


KNN
192
1
17
56
0
1
0
1
160
120
80
40