Applications of Data Analysis - Exercise 6

Tiina Nokelainen
503737

February 25, 2019

1 Part 1

1.1 Question 1.1

Why can't you trust the AUC result of Tux?

1.1.1 Answer 1.1

Tux trained his model with test data (he didn't split the data to training set and test set). The model is now trained really good for the tested data but it won't (probably) apply as well for other data.

1.2 Question 1.2

1. Does the high classification accuracy really mean that this is a good predictor?
2. Look at the test set predictions in p_test, what has this classifier actually learned?
3. What would the results look like if you used AUC instead of classification accuracy?

1.2.1 Answer 1.2

1. No it does not. It means that the model was able to predict our test set classes with 90 % accuracy.
2. Classifier has learnt that all are class "-1".
3. Approx 0.529 which is not very good score. It means that the model don't have good discrimination capacity to distinguish between positive class and negative clas.

2 Part 2: introduction to permutation tests

2.1 Question 2.1

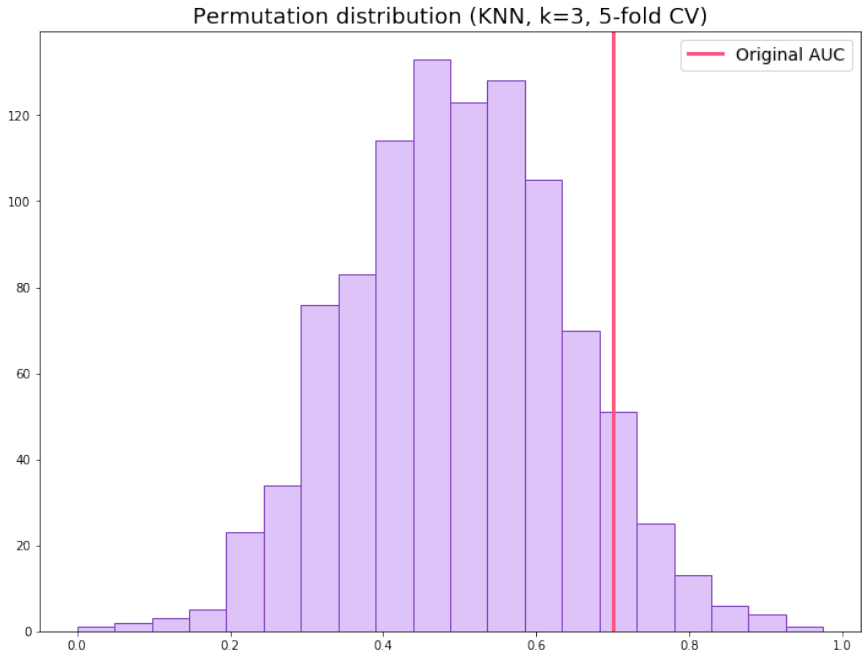
Implement a permutation test for the above analysis, are these results statistically significant with $\alpha = 0.05$? Provide both visualization of the permutation distribution, as well as the p-value.

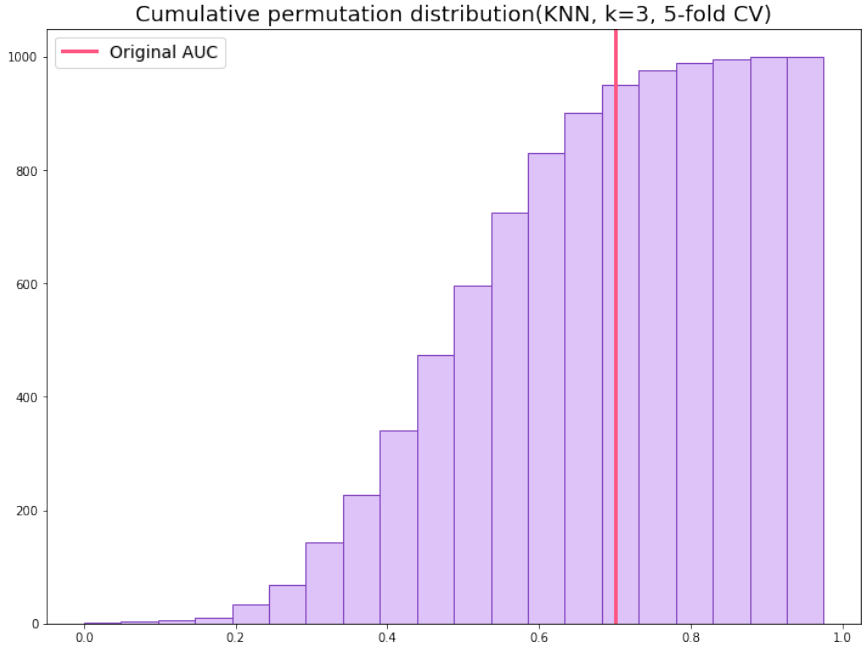
2.1.1 Answer 2.1

- Permutation test implemented below.
- The results are statistically significant since the p-value (= 0.1) was over $\alpha = 0.05$. This means that the in this case the predictions were more likely to be predicted by chance that by well trained model.

```
In [5]: import matplotlib.pyplot as plt
        size = 1000
        AUC_original = 0.7

In [6]: auc_perms = []
        for i in range(size):
            y = np.random.permutation(y)
            cv = StratifiedKFold(n_splits=5)
            cv_aucs = []
            for train, test in cv.split(X, y):
                X_train = X[train]
                y_train = y[train]
                X_test = X[test]
                y_test = y[test]
                knn = KNeighborsClassifier(n_neighbors=3)
                knn.fit(X_train, y_train)
                p_test = knn.predict_proba(X_test)[:,1]
                auc = roc_auc_score(y_test, p_test)
                cv_aucs.append(auc)
            auc_perms.append(np.mean(cv_aucs))

In [7]: counter = 0
        for pauc in auc_perms:
            if pauc >= AUC_original:
                counter += 1
        p_value = counter / size
        print("p-value is", p_value)

        plt.figure(figsize=(12,9))
        plt.hist(auc_perms, 20, color="#ddc3f7", ec="#8041bf")
        plt.title("Permutation distribution (KNN, k=3, 5-fold CV)", fontsize=18)
        plt.axvline(AUC_original, color="#ff547f", lw=3, label="Original AUC")
        plt.legend(fontsize=14)
        plt.show()

        plt.figure(figsize=(12,9))
        plt.hist(auc_perms, 20, color="#ddc3f7", ec="#8041bf", cumulative="true")
        plt.title("Cumulative permutation distribution(KNN, k=3, 5-fold CV)", fontsize=18)
        plt.axvline(AUC_original, color="#ff547f", lw=3, label="Original AUC")
        plt.legend(fontsize=14)
        plt.show()
```

```
p-value is 0.1
```


Permutation distribution (KNN, k=3, 5-fold CV)
Original AUC
120
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0


Cumulative permutation distribution(KNN, k=3, 5-fold CV)
Original AUC
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0

2.2 Question 2.2

Implement a permutation test for the above analysis, are these results statistically significant with $\alpha = 0.05$? Provide both visualization of the permutation distribution, as well as the p-value.

2.2.1 Answer 2.2

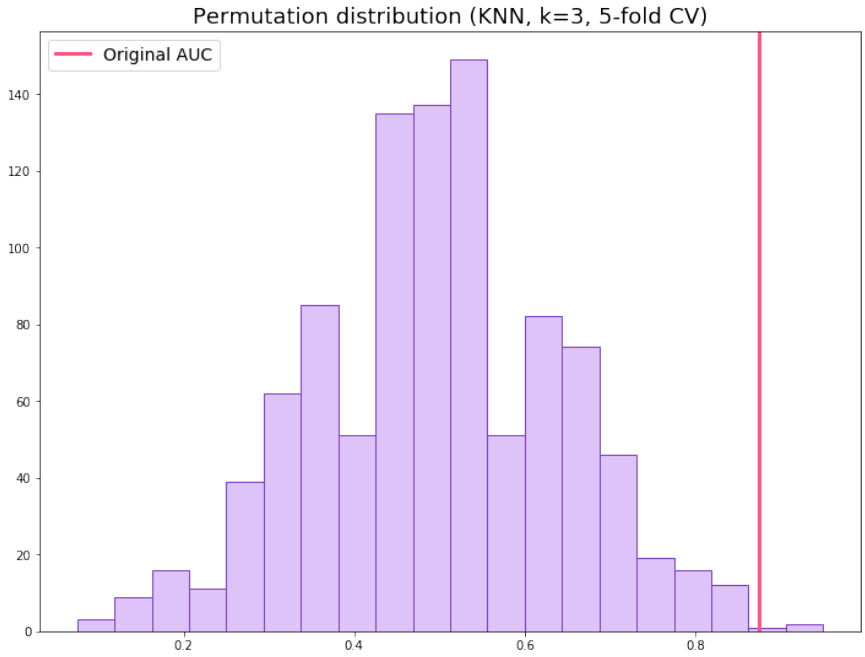
- Permutation test implementet below with hte visualizations
- The results are not statistically significant since the p-value (= 0.003) was lower than α.

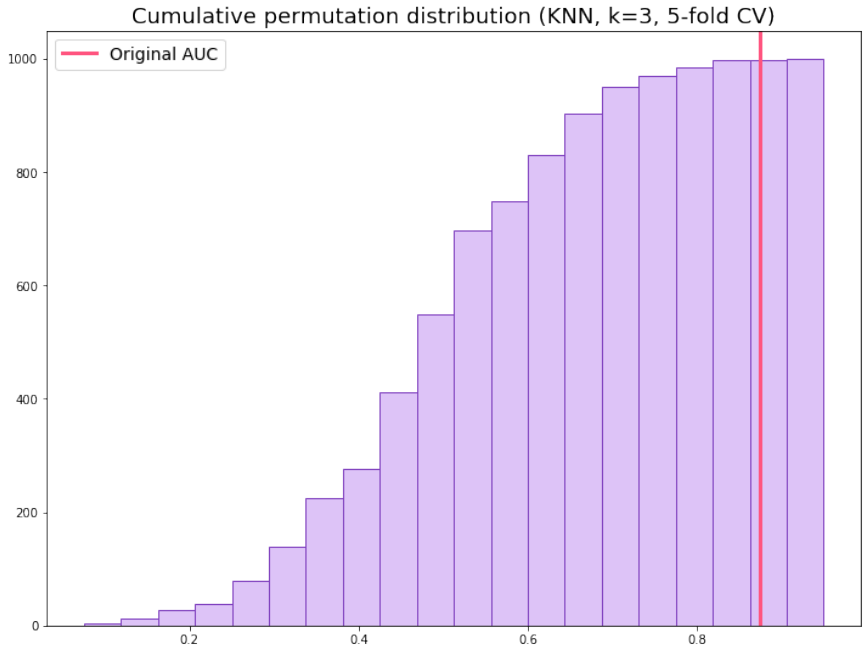
```
In [10]: AUC_original = 0.875
```

```
In [11]: auc_perms = []
         for i in range(size):
             y = np.random.permutation(y)
             cv = StratifiedKFold(n_splits=5)
             cv_aucs = []
             for train, test in cv.split(X, y):
                 X_train = X[train]
                 y_train = y[train]
                 X_test = X[test]
                 y_test = y[test]
                 knn = KNeighborsClassifier(n_neighbors=3)
                 knn.fit(X_train, y_train)
                 p_test = knn.predict_proba(X_test)[:,1]
                 auc = roc_auc_score(y_test, p_test)
                 cv_aucs.append(auc)
             auc_perms.append(np.mean(cv_aucs))
```

```
In [12]: counter = 0
         for pauc in auc_perms:
             if pauc >= AUC_original:
                 counter += 1
         p_value = counter / size
         print("p-value is", p_value)

         plt.figure(figsize=(12,9))
         plt.hist(auc_perms, 20, color="#ddc3f7", ec="#8041bf")
         plt.title("Permutation distribution (KNN, k=3, 5-fold CV)", fontsize=18)
         plt.axvline(AUC_original, color="#ff547f", lw=3, label="Original AUC")
         plt.legend(fontsize=14)
         plt.show()

         plt.figure(figsize=(12,9))
         plt.hist(auc_perms, 20,color="#ddc3f7", ec="#8041bf", cumulative="true")
         plt.title("Cumulative permutation distribution (KNN, k=3, 5-fold CV)", fontsize=18)
         plt.axvline(AUC_original, color="#ff547f", lw=3, label="Original AUC")
         plt.legend(fontsize=14)
         plt.show()
```

```
p-value is 0.003
```





3 Part 3: mis-using feature selection

3.1 Question 3.1

Use permutation test to show Tux that the feature selection based classification approach is actually not learning anything from the data ($\alpha = 0.05$, provide both visualization of the permutation distribution, as well as the p-value). Running the test may take a while. Analyse what is going on here, why did the results look so good?

3.1.1 Answers 3.1

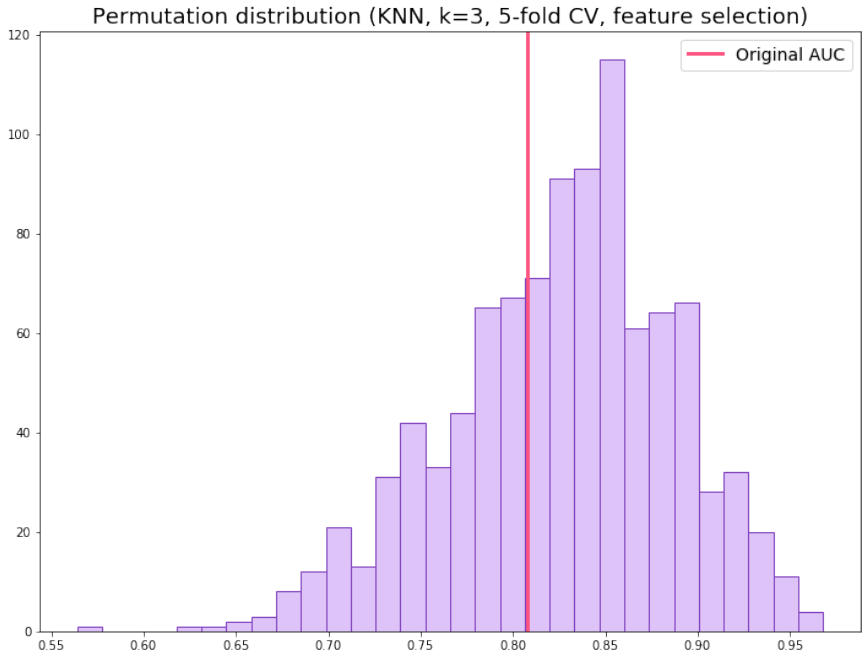
The feature selection should be part of the cross validation. Now the model has been validated for just one feature selection. The p-value is very high (0.656) so the results are statistically significant.

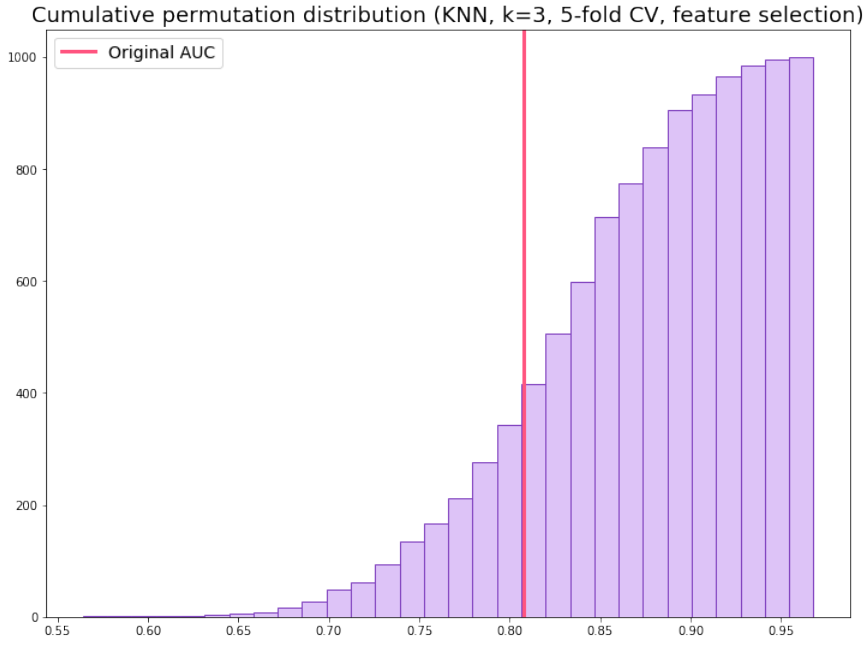
```
In [16]: auc_perms = []
         for i in range(1000):
             y = np.random.permutation(y)
             X_fs = select(X, y, 5)
             cv_aucs = []
             for train, test in cv.split(X_fs, y):
                 X_train = X_fs[train]
                 y_train = y[train]
                 X_test = X_fs[test]
                 y_test = y[test]
                 knn = KNeighborsClassifier(n_neighbors=3)
                 knn.fit(X_train, y_train)
                 p_test = knn.predict_proba(X_test)[:,1]
                 auc = roc_auc_score(y_test, p_test)
                 cv_aucs.append(auc)
             auc_perms.append(np.mean(cv_aucs))
```

```
In [17]: counter = 0
         for pauc in auc_perms:
             if pauc >= AUC_original:
                 counter += 1
         p_value = counter / size
         print("p-value is", p_value)

         plt.figure(figsize=(12,9))
         plt.hist(auc_perms, 30, color="#ddc3f7", ec="#8041bf")
         plt.title("Permutation distribution (KNN, k=3, 5-fold CV, feature selection)", fontsize
         plt.axvline(AUC_original, color="#ff547f", lw=3, label="Original AUC")
         plt.legend(fontsize=14)
         plt.show()

         plt.figure(figsize=(12,9))
         plt.hist(auc_perms, 30,color="#ddc3f7", ec="#8041bf", cumulative="true")
         plt.title("Cumulative permutation distribution (KNN, k=3, 5-fold CV, feature selection)
         plt.axvline(AUC_original, color="#ff547f", lw=3, label="Original AUC")
```

```
        plt.legend(fontsize=14)
        plt.show()
```

```
p-value is 0.656
```





3.2 Question 3.2 (bonus exercise)

Correct the bias in above example by combining feature selection properly with cross-validation, run the experiment again. Do also a permutation test for this experiment with as many permutations as you can afford in a reasonable amount of time.

In [20]:
```
cv = StratifiedKFold(n_splits=5)
cv_aucs = []
for train, test in cv.split(X, y):
    X_fs = select(X, y, 5)
    X_train = X_fs[train]
    y_train = y[train]
    X_test = X_fs[test]
    y_test = y[test]
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(X_train, y_train)
    p_test = knn.predict_proba(X_test)[:,1]
    auc = roc_auc_score(y_test, p_test)
    cv_aucs.append(auc)
cv_auc = np.mean(cv_aucs)

print(cv_auc)

auc_perms = []
for i in range(1000):
    y = np.random.permutation(y)
    cv_aucs = []
    for train, test in cv.split(X, y):
        X_fs = select(X, y, 5)
        X_train = X_fs[train]
        y_train = y[train]
        X_test = X_fs[test]
        y_test = y[test]
        knn = KNeighborsClassifier(n_neighbors=3)
        knn.fit(X_train, y_train)
        p_test = knn.predict_proba(X_test)[:,1]
        auc = roc_auc_score(y_test, p_test)
        cv_aucs.append(auc)
    auc_perms.append(np.mean(cv_aucs))
```

```
0.784
```

In [23]:
```
plt.figure(figsize=(12,9))
plt.hist(auc_perms, 30, color="#ddc3f7", ec="#8041bf")
plt.title("Permutation distribution (KNN, k=3, 5-fold CV with feature selection)", font
plt.show()

plt.figure(figsize=(12,9))
```

```
plt.hist(auc_perms, 30,color="#ddc3f7", ec="#8041bf", cumulative="true")
plt.title("Cumulative permutation distribution (KNN, k=3, 5-fold CV with feature select
plt.show()
```