Internship Report

PREDICTING KIDNEY CANCER SURVIVAL FROM GENOMIC DATA

BACHELOR OF TECHNOLOGY COMPUTER SCIENCE ENGINEERING

SUBMITTED BY

Kartik Tyagi (CSE with specialization in OSOS, R100218024)

Sahil Sangwan (CSE with specialization in OSOS, R100218050)

Shubham Patel (CSE with specialization in OSOS, R100216088)

Divyanshu Sharma (CSE with specialization in Graphics and Gaming, R142218026)

Parixit Tomar (CSE with specialization in IOT and Smart Cities, R164218104)

GUIDED BY: Abhijit Kumar

Table of Contents

Executive Summary

Using bioinformatics approaches to identify genes that are useful for the diagnosis and prognosis prediction of patients with cancer. The analysis of cancer data is important yet difficult due to the large amounts of gene expression data available. Only features that can talk about the health condition of patients must be extracted. The development of efficient classification models based on the extracted data (i.e., genes) is helpful for early diagnosis and prognosis prediction of patients with cancer. Cancer is caused by gene modifications. Here we would extract genes useful for the prognosis prediction of patients with kidney cancer and then predicted prognosis by applying a classification algorithm based on the gene. Kidney cancer is a primary tumour generated from the kidney, among which malignant renal cell carcinoma accounts for over 90% of cases. As kidney cancer shows no symptoms at the early stages it is often diagnosed at a final stage. According to registered statistics for cancer in Korea, 5043 kidney cancer cases were diagnosed in 2016. There have been various successful applications of machine learning and data mining techniques to bioinformatics and genomics research. PathAI was implemented for digital pathology after the analysis of image data from patients with breast cancer using AI decreasing the error rate of diagnosing metastasized cancer with the help of deep learning. Over the years, various technologies for data mining have been applied for predicting various diseases and other problems. Due to the advantages of deep learning various deep learning approaches have been applied for the prediction of cancer using gene expression data. Deep learning approaches are useful for constructing predictive models and feature extraction.

1. Background

Aim: Given 24 health related attributes taken in 2-month period of 400 patients, using the information of the 158 patients with complete records to predict the outcome (i.e., whether one has chronic kidney disease) of the remaining 242 patients (with missing values in their records).

2. System

2.1 Requirements

Hardware:

- RAM: 2GB
- Disk Space: 4GB

Software:

- Python IDE

Operating System:

- Windows.

2.2 Design, Architecture, Implementation and Testing

Modules and helper functions

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import roc_curve, auc, confusion_matrix, classification_report,accuracy_sc
ore
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

def auc_scorer(clf, X, y, model): # Helper function to plot the ROC curve
    if model=='RF':
        fpr, tpr, _ = roc_curve(y, clf.predict_proba(X)[:,1])
    elif model=='SVM':
        fpr, tpr, _ = roc_curve(y, clf.decision_function(X))
    roc_auc = auc(fpr, tpr)

    plt.figure()    # Plot the ROC curve
    plt.plot(fpr, tpr, label='ROC curve from '+model+' model (area = %0.3f)' % roc_auc)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend(loc="lower right")
    plt.show()

    return fpr,tpr,roc_auc

# from subprocess import check_output
# print(check_output(["ls", "../input"]).decode("utf8"))
```

Loading Files

```
df = pd.read_csv('../input/kidney_disease.csv')
```

Cleaning and Pre-processing of Data for Training a Classifier

```
# Map text to 1/0 and do some cleaning
df[['htn','dm','cad','pe','ane']] = df[['htn','dm','cad','pe','ane']].replace(to_replace={'ye
s':1,'no':0})
df[['rbc','pc']] = df[['rbc','pc']].replace(to_replace={'abnormal':1,'normal':0})
df[['pcc','ba']] = df[['pcc','ba']].replace(to_replace={'present':1,'notpresent':0})
df[['appet']] = df[['appet']].replace(to_replace={'good':1,'poor':0,'no':np.nan})
df['classification'] = df['classification'].replace(to_replace={'ckd':1.0,'ckd\t':1.0,'notckd':
0.0,'no':0.0})
df.rename(columns={'classification':'class'},inplace=True)
```

```
# Further cleaning
df['pe'] = df['pe'].replace(to_replace='good',value=0) # Not having pedal edema is good
df['appet'] = df['appet'].replace(to_replace='no',value=0)
df['cad'] = df['cad'].replace(to_replace='\tno',value=0)
df['dm'] = df['dm'].replace(to_replace={'\tno':0,'\tyes':1,' yes':1, '':np.nan})
df.drop('id',axis=1,inplace=True)
```

```
df.head()
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe
0	48.0	80.0	1.020	1.0	0.0	NaN	0.0	0.0	0.0	121.0	...	44	7800	5.2	1.0	1.0	0.0	1.0	0.0
1	7.0	50.0	1.020	4.0	0.0	NaN	0.0	0.0	0.0	NaN	...	38	6000	NaN	0.0	0.0	0.0	1.0	0.0
2	62.0	80.0	1.010	2.0	3.0	0.0	0.0	0.0	0.0	423.0	...	31	7500	NaN	0.0	1.0	0.0	0.0	0.0
3	48.0	70.0	1.005	4.0	0.0	0.0	1.0	1.0	0.0	117.0	...	32	6700	3.9	1.0	0.0	0.0	0.0	1.0
4	51.0	80.0	1.010	2.0	0.0	0.0	0.0	0.0	0.0	106.0	...	35	7300	4.6	0.0	0.0	0.0	1.0	0.0

Checking the portion of rows with NaN

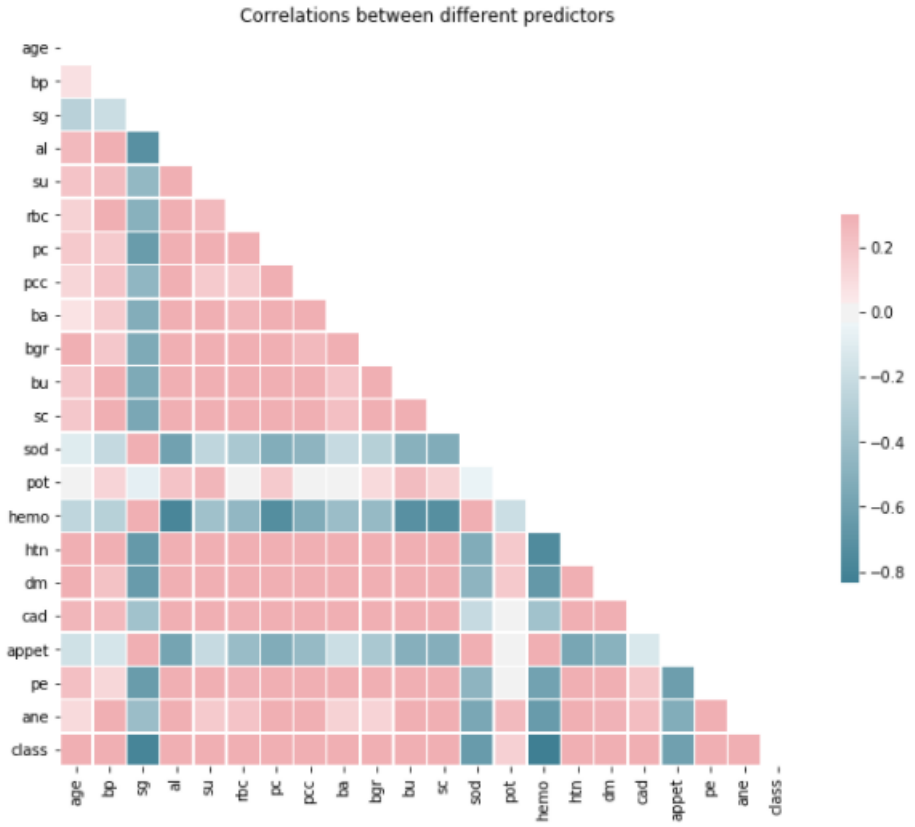
- Now the data is cleaned with improper values labelled NaN. Let's see how many NaNs are there.
- Drop all the rows with NaN values, and build a model out of this dataset (i.e., df2)

```
df2 = df.dropna(axis=0)
df2['class'].value_counts()
```

```
0.0    115
1.0     43
Name: class, dtype: int64
```

Examining Correlations Between Different Features

```
corr_df = df2.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr_df, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr_df, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Correlations between different predictors')
plt.show()
```



Splitting the Set for Training Models further into a (sub-)training set and testing set

```
X_train, X_test, y_train, y_test = train_test_split(df2.iloc[:,:-1], df2['class'],
                                                    test_size = 0.33, random_state=44,
                                                   stratify= df2['class'] )
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(105, 24)
(53, 24)
```

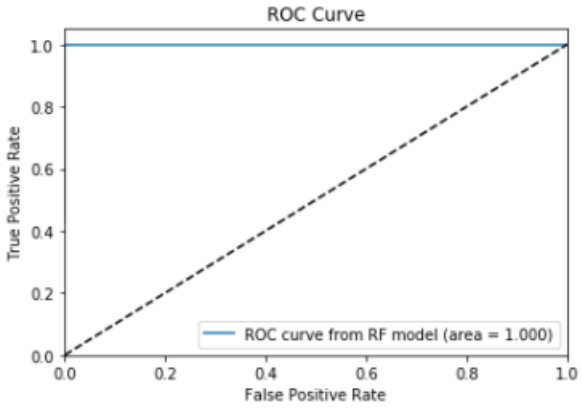
```
y_train.value_counts()
```

```
0.0    76
1.0    29
Name: class, dtype: int64
```

Choosing parameters with GridSearchCV with 10-fold cross validations.

```
tuned_parameters = [{'n_estimators':[7,8,9,10,11,12,13,14,15,16],'max_depth':[2,3,4,5,6,None],
                     'class_weight':[None,{0: 0.33,1:0.67},'balanced'],'random_state':[42]}]
clf = GridSearchCV(RandomForestClassifier(), tuned_parameters, cv=10,scoring='f1')
clf.fit(X_train, y_train)

print("Detailed classification report:")
y_true, lr_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, lr_pred))

confusion = confusion_matrix(y_test, lr_pred)
print('Confusion Matrix:')
print(confusion)

# Determine the false positive and true positive rates
fpr,tpr,roc_auc = auc_scorer(clf, X_test, y_test, 'RF')

print('Best parameters:')
print(clf.best_params_)
clf_best = clf.best_estimator_
```

```
Detailed classification report:
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00        39
        1.0       1.00      1.00      1.00        14

avg / total       1.00      1.00      1.00        53

Confusion Matrix:
[[39  0]
 [ 0 14]]
```

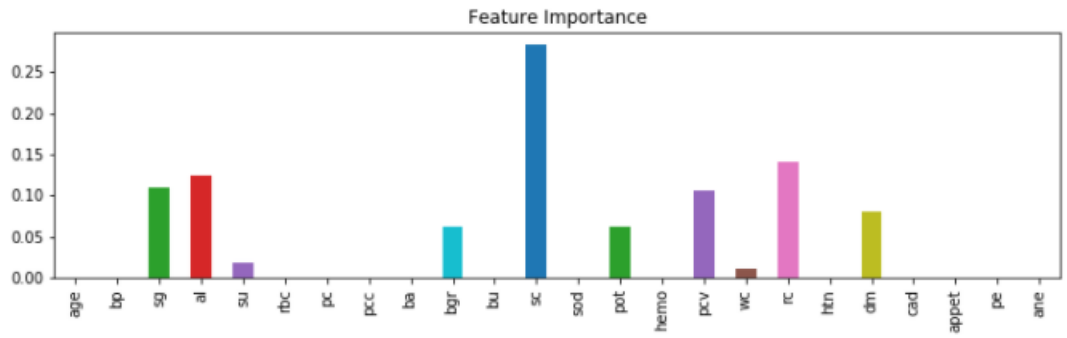



```
Best parameters:
{'class_weight': None, 'max_depth': 2, 'n_estimators': 8, 'random_state': 42}
```

Examine feature importance

```
plt.figure(figsize=(12,3))
features = X_test.columns.values.tolist()
importance = clf_best.feature_importances_.tolist()
feature_series = pd.Series(data=importance,index=features)
feature_series.plot.bar()
plt.title('Feature Importance')
```

```
Text(0.5,1,'Feature Importance')
```

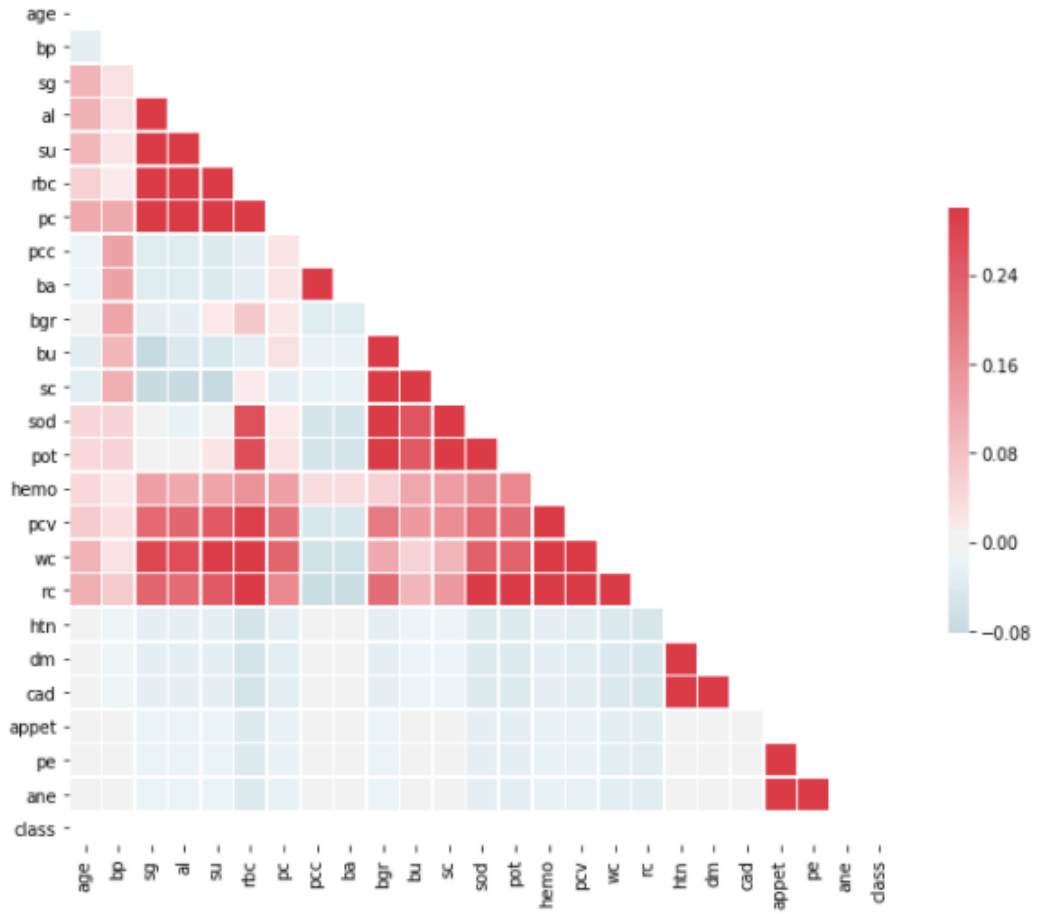



```
list_to_fill = X_test.columns[feature_series>0]
print(list_to_fill)
```

```
Index(['sg', 'al', 'su', 'bgr', 'sc', 'pot', 'pcv', 'wc', 'rc', 'dm'], dtype='object')
```

Examining the rest of the dataset

```
# Are there correlation in missing values?
corr_df = pd.isnull(df).corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr_df, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr_df, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.show()
```



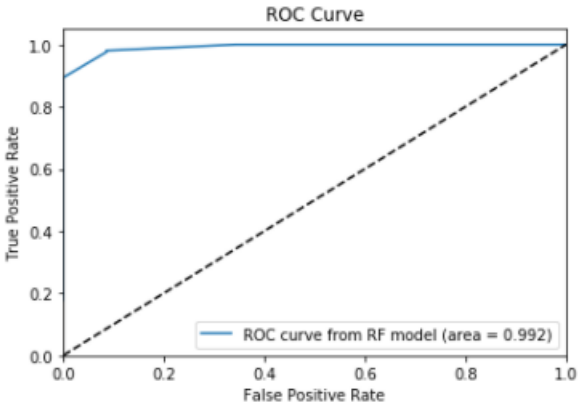
Making predictions with the best model selected above:

We filled in all NaN with 0 and pass it to the trained classifier. The results are as follows:

- True positive = 180
- True negative = 35
- False positive = 0
- False negative = 27
- Accuracy = 88.8%
- ROC AUC = 99.2%

```
df2 = df.dropna(axis=0)
no_na = df2.index.tolist()
some_na = df.drop(no_na).apply(lambda x: pd.to_numeric(x,errors='coerce'))
some_na = some_na.fillna(0) # Fill up all Nan by zero.

X_test = some_na.iloc[:,:-1]
y_test = some_na['class']
y_true = y_test
lr_pred = clf_best.predict(X_test)
print(classification_report(y_true, lr_pred))

confusion = confusion_matrix(y_test, lr_pred)
print('Confusion Matrix:')
print(confusion)

print('Accuracy: %3f' % accuracy_score(y_true, lr_pred))
# Determine the false positive and true positive rates
fpr,tpr,roc_auc = auc_scorer(clf_best, X_test, y_test, 'RF')
```

```
             precision    recall  f1-score   support

        0.0       0.56      1.00      0.72        35
        1.0       1.00      0.87      0.93       207

avg / total       0.94      0.89      0.90       242

Confusion Matrix:
[[ 35   0]
 [ 27 180]]
Accuracy: 0.888430
```



3.Conclusion

With proper tuning of parameters using cross-validation in the training set, the Random Forest Classifier achieves an accuracy of 88.8% and an ROC AUC of 99.2%. Lesson learnt: It happens that some pruning helps improve the performance of RF a lot.

5. References

[1] Villarroel, Maria Central.” Personalizing cancer treatment in the age of global genomic analyses: PALB2 gene mutations and the response to DNA damaging agents in pan- cretic cancer.” Molecular cancer therapeutics 10.1 (2011): 3-8.

[2] Chow W-H, Dong LM, Devesa SS. Epidemiology and risk factors for kidney cancer. Nature reviews. Urology 2010;7(5):245-257: 10.1038/nrurol.2010.46.

[3] Pirooznia M, Yang JY, Yang MQ, Deng Y. A comparative study of different machine learning methods on microarray gene expression data. BMC Genomics. 2008;9 Supply 1: S13.

[4] Loeb LA, Loeb KR, Anderson JP. Multiple mutations and cancer. Proc Natl Acad Sci USA. 2003;100(3):776-81.

[5] Nowell PC. The clonal evolution of tumour cell populations. Science. 1976;194(4260):23-28.

- The results shown here are in whole or part based upon data generated by the TCGA Research Network: http://cancergenome.nih.gov/.
- Cancer Genome Atlas Network.” Comprehensive molecular portraits of human breast tumours.” Nature 490.7418 (2012): 61-70.
- Gross, Andrew Metal. ”Multi-tiered genomic analysis of head and neck cancer ties TP53 mutation to 3p loss.” Nature genetics (2014).
- Rossi, Davide, Metal.” Mutations of NOTCH1 are an in- dependent predictor of survival in chronic lymphocytic leukaemia.” Blood 119.2 (2012): 521-529.