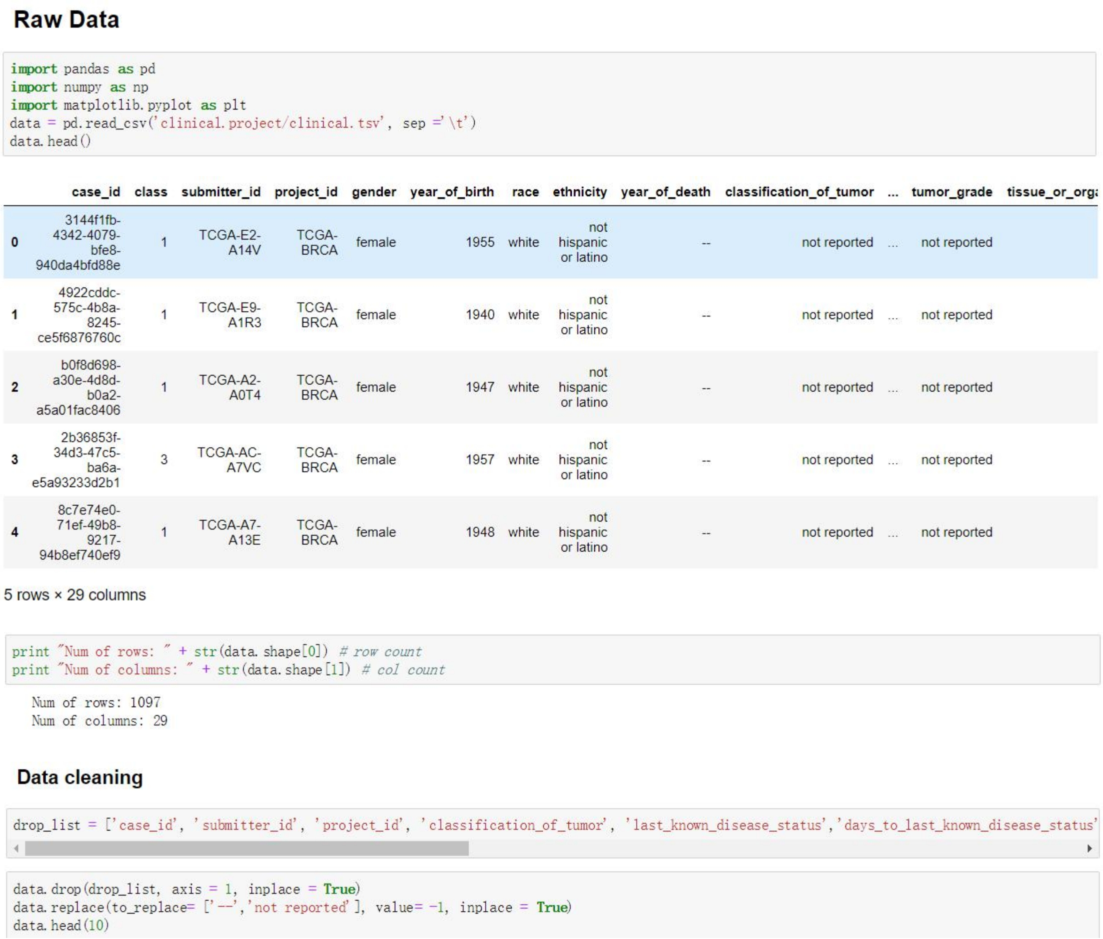
A Machine Learning Project

No.1 : Data Exploration

- You can find my Data on https://portal.gdc.cancer.gov/projects/TCGA-BRCA



Raw Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('clinical.project/clinical.tsv', sep ='\t')
data.head()
```

	case_id	class	submitter_id	project_id	gender	year_of_birth	race	ethnicity	year_of_death	classification_of_tumor	...	tumor_grade	tissue_or_orga
0	3144f1fb-4342-4079-bfe8-940da4bfd88e	1	TCGA-E2-A14V	TCGA-BRCA	female	1955	white	not hispanic or latino	--	not reported	...	not reported	
1	4922cddc-575c-4b8a-8245-ce5f6876760c	1	TCGA-E9-A1R3	TCGA-BRCA	female	1940	white	not hispanic or latino	--	not reported	...	not reported	
2	b0f8d698-a30e-4d8d-b0a2-a5a01fac8406	1	TCGA-A2-A0T4	TCGA-BRCA	female	1947	white	not hispanic or latino	--	not reported	...	not reported	
3	2b36853f-34d3-47c5-ba6a-e5a93233d2b1	3	TCGA-AC-A7VC	TCGA-BRCA	female	1957	white	not hispanic or latino	--	not reported	...	not reported	
4	8c7e74e0-71ef-49b8-9217-94b8ef740ef9	1	TCGA-A7-A13E	TCGA-BRCA	female	1948	white	not hispanic or latino	--	not reported	...	not reported	

5 rows × 29 columns

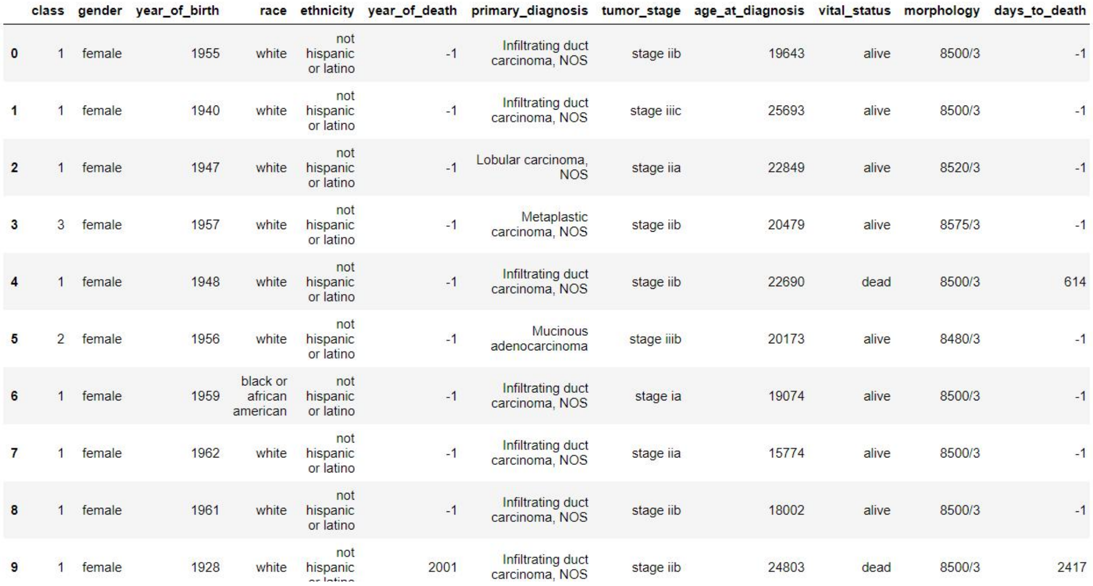
```
print "Num of rows: " + str(data.shape[0]) # row count
print "Num of columns: " + str(data.shape[1]) # col count
```

```
Num of rows: 1097
Num of columns: 29
```

Data cleaning

```
drop_list = ['case_id', 'submitter_id', 'project_id', 'classification_of_tumor', 'last_known_disease_status','days_to_last_known_disease_status'
```

```
data.drop(drop_list, axis = 1, inplace = True)
data.replace(to_replace= ['--','not reported'], value= -1, inplace = True)
data.head(10)
data.drop(drop_list, axis = 1, inplace = True)
data.replace(to_replace= ['--','not reported'], value= -1, inplace = True)
data.head(10)
```

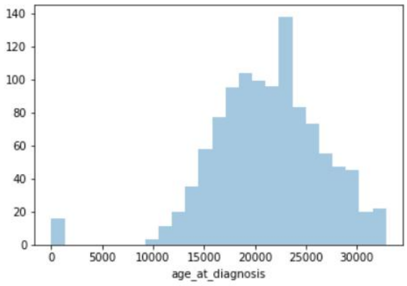


	class	gender	year_of_birth	race	ethnicity	year_of_death	primary_diagnosis	tumor_stage	age_at_diagnosis	vital_status	morphology	days_to_death
0	1	female	1955	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iib	19643	alive	8500/3	-1
1	1	female	1940	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iiic	25693	alive	8500/3	-1
2	1	female	1947	white	not hispanic or latino	-1	Lobular carcinoma, NOS	stage iia	22849	alive	8520/3	-1
3	3	female	1957	white	not hispanic or latino	-1	Metaplastic carcinoma, NOS	stage iib	20479	alive	8575/3	-1
4	1	female	1948	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iib	22690	dead	8500/3	614
5	2	female	1956	white	not hispanic or latino	-1	Mucinous adenocarcinoma	stage iiib	20173	alive	8480/3	-1
6	1	female	1959	black or african american	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage ia	19074	alive	8500/3	-1
7	1	female	1962	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iia	15774	alive	8500/3	-1
8	1	female	1961	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iib	18002	alive	8500/3	-1
9	1	female	1928	white	not hispanic	2001	Infiltrating duct carcinoma, NOS	stage iib	24803	dead	8500/3	2417

Plot

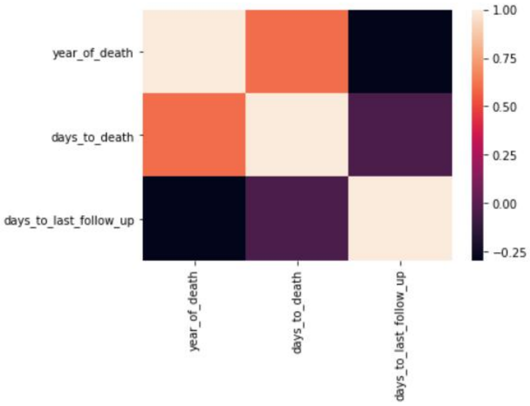
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sb

sb.distplot(data['age_at_diagnosis'].map(lambda x: float(x)), kde=False)
```

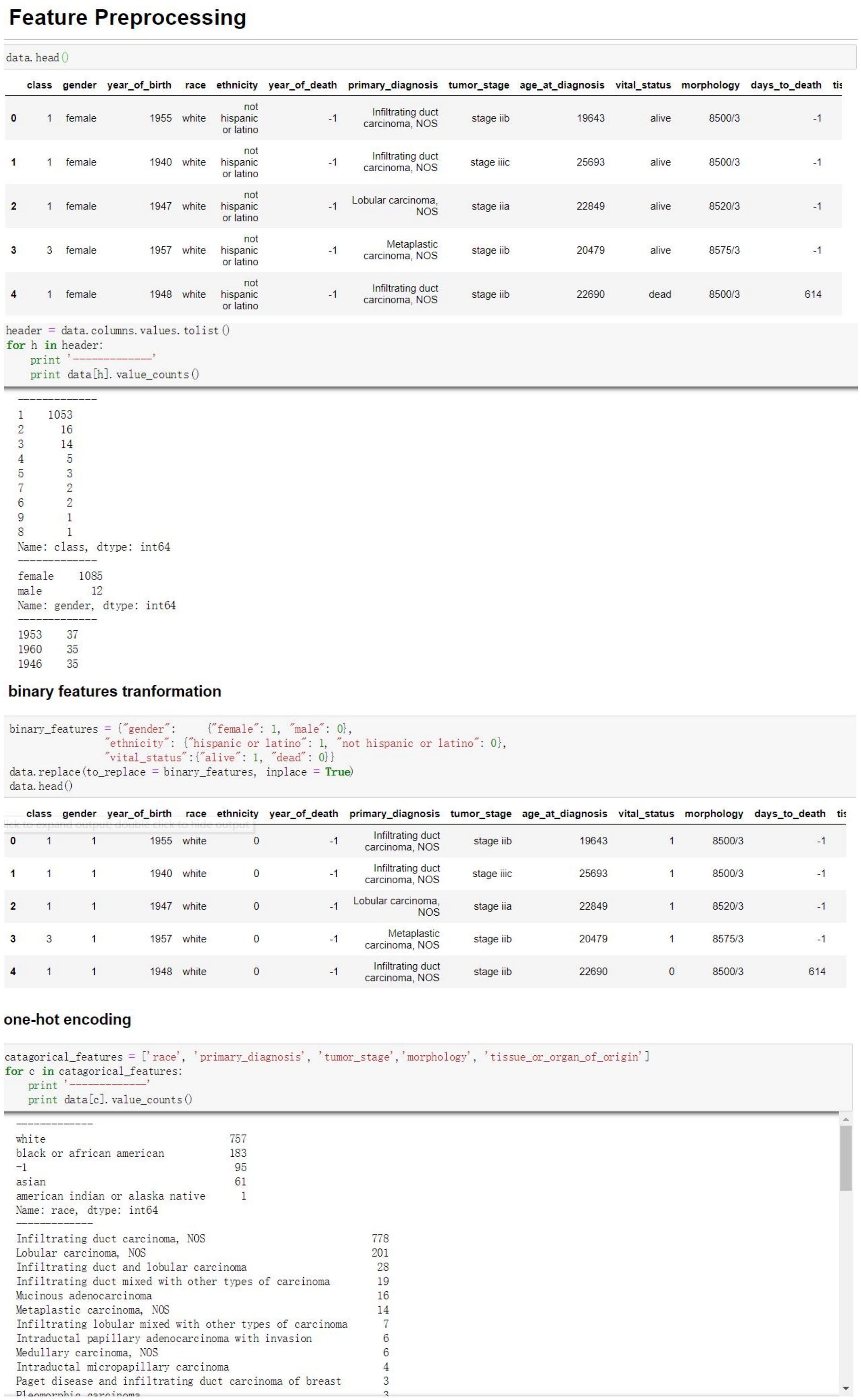
```
<matplotlib.axes._subplots.AxesSubplot at 0x119585310>
```



```
corr = data[["year_of_death","days_to_death","days_to_last_follow_up"
             ]].astype(int).corr()
sb.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1197c4c90>
```





Feature Preprocessing

```
data.head()
```

	class	gender	year_of_birth	race	ethnicity	year_of_death	primary_diagnosis	tumor_stage	age_at_diagnosis	vital_status	morphology	days_to_death	tis
0	1	female	1955	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iib	19643	alive	8500/3	-1	
1	1	female	1940	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iiic	25693	alive	8500/3	-1	
2	1	female	1947	white	not hispanic or latino	-1	Lobular carcinoma, NOS	stage iia	22849	alive	8520/3	-1	
3	3	female	1957	white	not hispanic or latino	-1	Metaplastic carcinoma, NOS	stage iib	20479	alive	8575/3	-1	
4	1	female	1948	white	not hispanic or latino	-1	Infiltrating duct carcinoma, NOS	stage iib	22690	dead	8500/3	614	

```
header = data.columns.values.tolist()
for h in header:
    print '-------------'
    print data[h].value_counts()
```

```
-------------
1    1053
2      16
3      14
4       5
5       3
7       2
6       2
9       1
8       1
Name: class, dtype: int64
-------------
female    1085
male        12
Name: gender, dtype: int64
-------------
1953    37
1960    35
1946    35
```

binary features tranformation

```
binary_features = {"gender":      {"female": 1, "male": 0},
                "ethnicity": {"hispanic or latino": 1, "not hispanic or latino": 0},
                "vital_status":{"alive": 1, "dead": 0}}
data.replace(to_replace = binary_features, inplace = True)
data.head()
```

	class	gender	year_of_birth	race	ethnicity	year_of_death	primary_diagnosis	tumor_stage	age_at_diagnosis	vital_status	morphology	days_to_death	tis
0	1	1	1955	white	0	-1	Infiltrating duct carcinoma, NOS	stage iib	19643	1	8500/3	-1	
1	1	1	1940	white	0	-1	Infiltrating duct carcinoma, NOS	stage iiic	25693	1	8500/3	-1	
2	1	1	1947	white	0	-1	Lobular carcinoma, NOS	stage iia	22849	1	8520/3	-1	
3	3	1	1957	white	0	-1	Metaplastic carcinoma, NOS	stage iib	20479	1	8575/3	-1	
4	1	1	1948	white	0	-1	Infiltrating duct carcinoma, NOS	stage iib	22690	0	8500/3	614	

one-hot encoding

```
catagorical_features = ['race', 'primary_diagnosis', 'tumor_stage','morphology', 'tissue_or_organ_of_origin']
for c in catagorical_features:
    print '-------------'
    print data[c].value_counts()
```

```
-------------
white                               757
black or african american           183
-1                                   95
asian                                61
american indian or alaska native      1
Name: race, dtype: int64
-------------
Infiltrating duct carcinoma, NOS                        778
Lobular carcinoma, NOS                                  201
Infiltrating duct and lobular carcinoma                  28
Infiltrating duct mixed with other types of carcinoma    19
Mucinous adenocarcinoma                                  16
Metaplastic carcinoma, NOS                               14
Infiltrating lobular mixed with other types of carcinoma  7
Intraductal papillary adenocarcinoma with invasion        6
Medullary carcinoma, NOS                                  6
Intraductal micropapillary carcinoma                      4
Paget disease and infiltrating duct carcinoma of breast   3
Pleomorphic carcinoma                                     3
```

```
all_features = data.columns.values.tolist()
res_features = [item for item in all_features if item not in catagorical_features]
data_processed = pd.get_dummies(data, columns = catagorical_features).astype(int)
```

Analysis and find the correlation matix

```
corr
```

	class	gender	year_of_birth	ethnicity	year_of_death	age_at_diagnosis	vital_status	days_to_death	days_to_last
class	1.000000	0.017379	-0.025935	0.055296	-0.006386	0.050816	-0.004586	-0.005895	
gender	0.017379	1.000000	-0.002229	-0.009875	0.034034	-0.034024	-0.016811	0.027792	
year_of_birth	-0.025935	-0.002229	1.000000	0.080714	-0.013836	-0.030576	0.060082	-0.054882	
ethnicity	0.055296	-0.009875	0.080714	1.000000	0.035199	-0.103388	-0.060524	0.051049	
year_of_death	-0.006386	0.034034	-0.013836	0.035199	1.000000	0.069480	-0.806928	0.602938	
age_at_diagnosis	0.050816	-0.034024	-0.030576	-0.103388	0.069480	1.000000	-0.091163	0.030538	
vital_status	-0.004586	-0.016811	0.060082	-0.060524	-0.806928	-0.091163	1.000000	-0.755990	
days_to_death	-0.005895	0.027792	-0.054882	0.051049	0.602938	0.030538	-0.755990	1.000000	
days_to_last_follow_up	-0.018385	0.005089	0.023476	0.082379	-0.296280	-0.140830	0.225141	-0.041420	
race_-1	-0.050883	0.001221	-0.125751	-0.592219	-0.044208	0.143775	0.057820	-0.054318	
race_american indian or alaska native	-0.004992	0.003177	0.002918	0.008786	-0.009775	-0.005782	0.012114	-0.009158	

Modeling

k-fold cross validation (k=5)

```
from sklearn.cross_validation import KFold
def run_cv(X,y,clf_class,**kwargs):
    kf = KFold(len(y),n_folds=5,shuffle=False)
    y_pred = y.copy()
    clf = clf_class(**kwargs)
    for train_index, test_index in kf:
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train = y.iloc[train_index]
        clf.fit(X_train,y_train)
        y_pred[test_index] = clf.predict(X_test)
    return y_pred
```

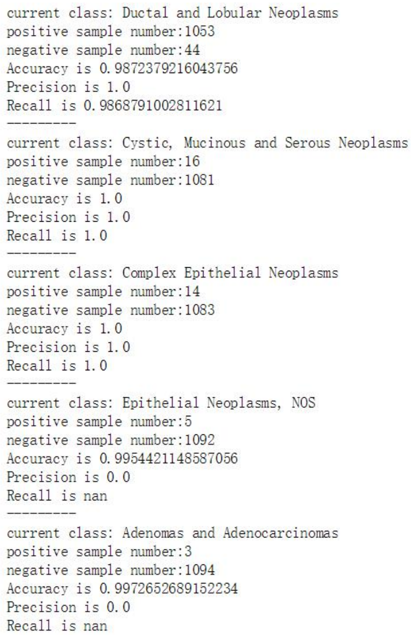
Supervised Learning Models

```
import xgboost as xgb
from xgboost import XGBClassifier

def accuracy(y_true,y_pred):
    return np.mean(y_true == y_pred)

features = data_processed.columns.tolist()
X = data_processed[features[1:]]
y = data_processed[features[0]]

xgboost_result = run_cv(X = X,  y = y,  clf_class= XGBClassifier,  objective = 'multi:softmax',  num_class = 9)
print 'xgboost accuracy:' + str(accuracy(y,  xgboost_result))
```

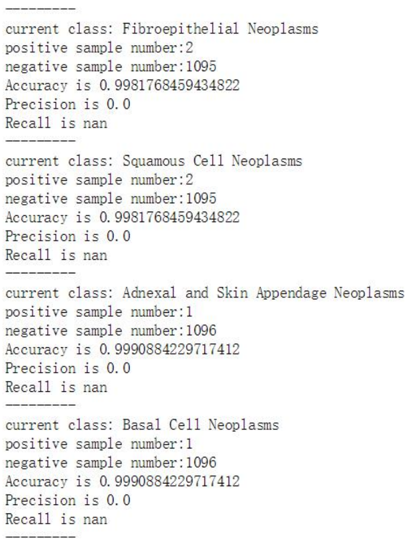
```
xgboost accuracy:0.9872379216043756
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from collections import Counter

def cal_evaluation(cm):
    tn = cm[0][0]
    fp = cm[0][1]
    fn = cm[1][0]
    tp = cm[1][1]
    accuracy  = (tp + tn) / (tp + fp + fn + tn + 0.0)
    precision = tp / (tp + fp + 0.0)
    recall = tp / (tp + fn + 0.0)
    print "Accuracy is " + str(accuracy)
    print "Precision is " + str(precision)
    print "Recall is " + str(recall)

class_list = ["Ductal and Lobular Neoplasms", "Cystic, Mucinous and Serous Neoplasms", "Complex Epithelial Neoplasms",
              "Epithelial Neoplasms, NOS", "Adenomas and Adenocarcinomas", "Fibroepithelial Neoplasms",
              "Squamous Cell Neoplasms", "Adnexal and Skin Appendage Neoplasms", "Basal Cell Neoplasms"]
for i in range(1,10):
    class_idx = i
    result_i = map(lambda a : 1 if a == i else 0,  xgboost_result[:])
    y_i = map(lambda b : 1 if b == i else 0,  y[:])
    print 'current class: ' + class_list[i-1]
    print 'positive sample number:' + str(Counter(y_i)[1])
    print 'negative sample number:' + str(Counter(y_i)[0])
    cal_evaluation(confusion_matrix(result_i,  y_i))
    print '---------'
```

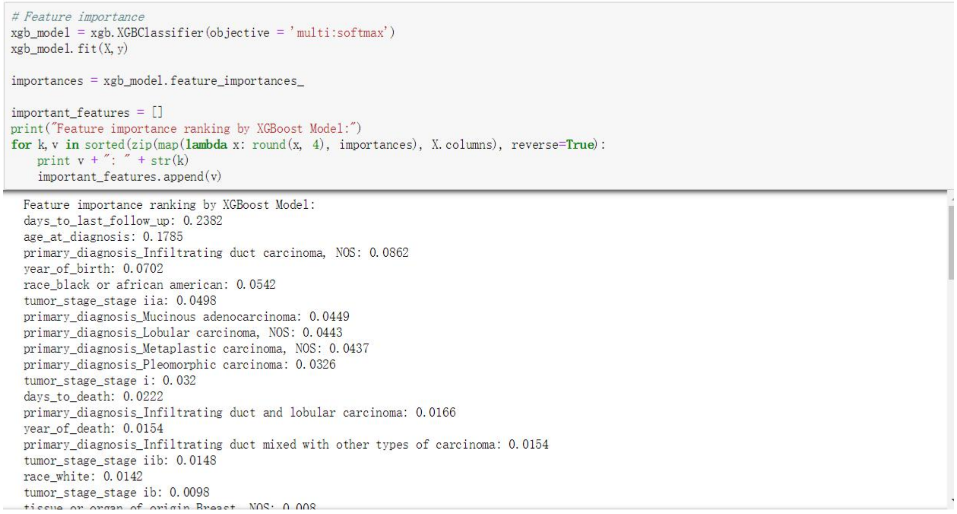


```
current class: Ductal and Lobular Neoplasms
positive sample number:1053
negative sample number:44
Accuracy is 0.9872379216043756
Precision is 1.0
Recall is 0.9868791002811621
---------
current class: Cystic, Mucinous and Serous Neoplasms
positive sample number:16
negative sample number:1081
Accuracy is 1.0
Precision is 1.0
Recall is 1.0
---------
current class: Complex Epithelial Neoplasms
positive sample number:14
negative sample number:1083
Accuracy is 1.0
Precision is 1.0
Recall is 1.0
---------
current class: Epithelial Neoplasms, NOS
positive sample number:5
negative sample number:1092
Accuracy is 0.9954421148587056
Precision is 0.0
Recall is nan
---------
current class: Adenomas and Adenocarcinomas
positive sample number:3
negative sample number:1094
Accuracy is 0.9972652689152234
Precision is 0.0
Recall is nan
```



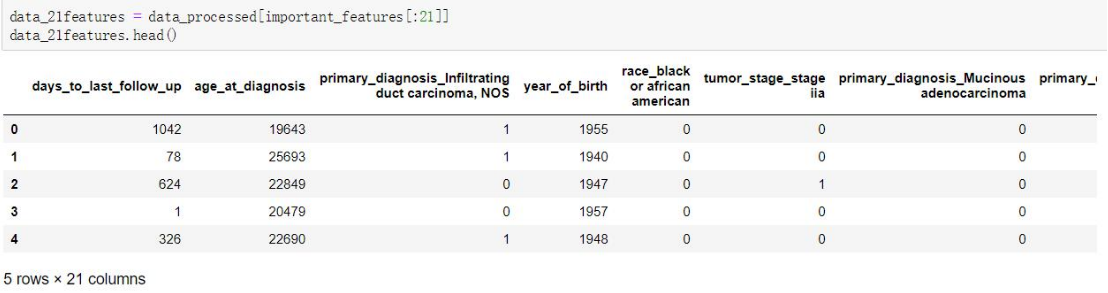
```
---------
current class: Fibroepithelial Neoplasms
positive sample number:2
negative sample number:1095
Accuracy is 0.9981768459434822
Precision is 0.0
Recall is nan
---------
current class: Squamous Cell Neoplasms
positive sample number:2
negative sample number:1095
Accuracy is 0.9981768459434822
Precision is 0.0
Recall is nan
---------
current class: Adnexal and Skin Appendage Neoplasms
positive sample number:1
negative sample number:1096
Accuracy is 0.9990884229717412
Precision is 0.0
Recall is nan
---------
current class: Basal Cell Neoplasms
positive sample number:1
negative sample number:1096
Accuracy is 0.9990884229717412
Precision is 0.0
Recall is nan
---------
```

Feature selection



```
# Feature importance
xgb_model = xgb.XGBClassifier(objective = 'multi:softmax')
xgb_model.fit(X,y)

importances = xgb_model.feature_importances_

important_features = []
print("Feature importance ranking by XGBoost Model:")
for k,v in sorted(zip(map(lambda x: round(x, 4), importances), X.columns), reverse=True):
    print v + ": " + str(k)
    important_features.append(v)
```

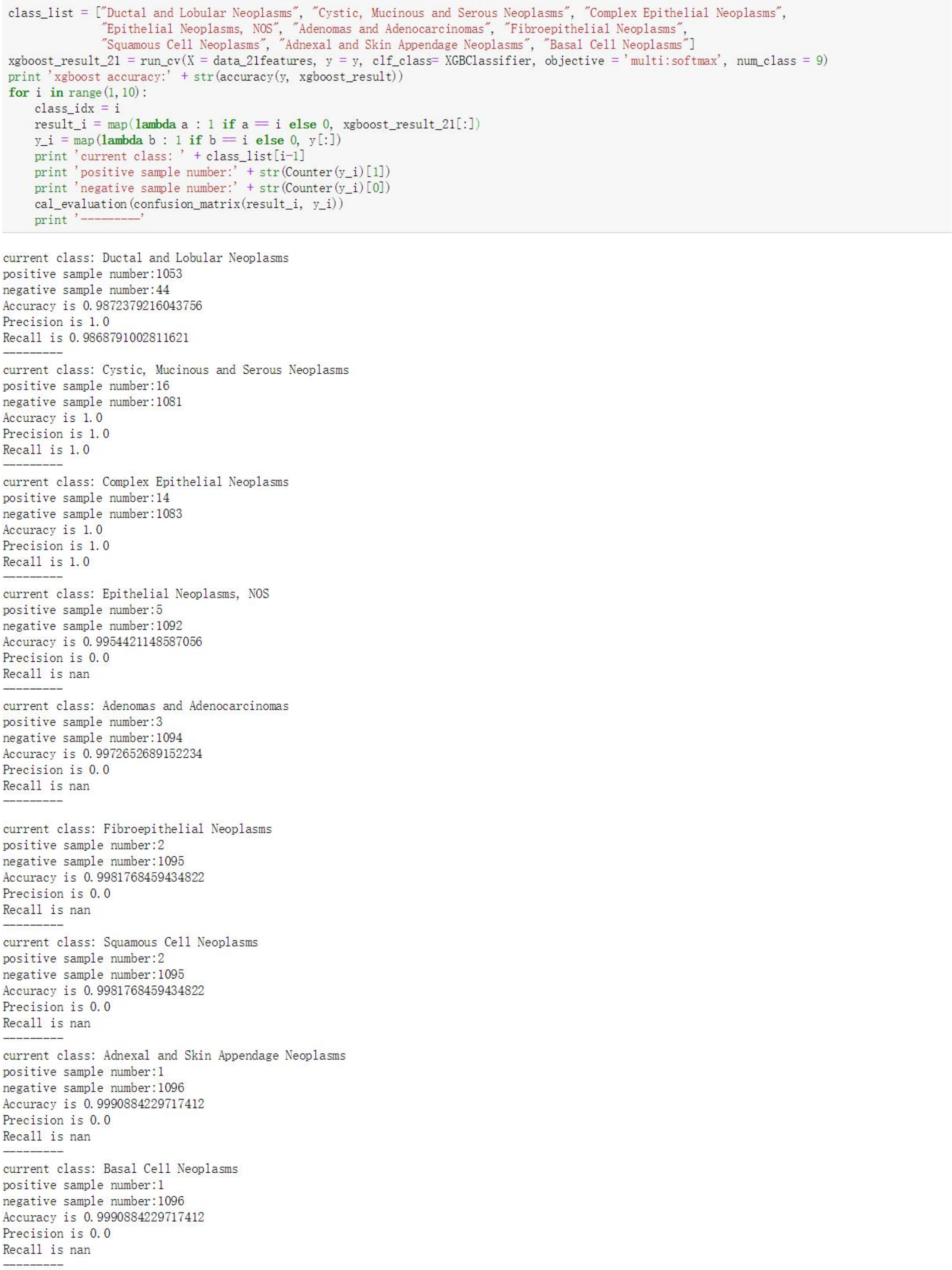
```
Feature importance ranking by XGBoost Model:
days_to_last_follow_up: 0.2382
age_at_diagnosis: 0.1785
primary_diagnosis_Infiltrating duct carcinoma, NOS: 0.0862
year_of_birth: 0.0702
race_black or african american: 0.0542
tumor_stage_stage iia: 0.0498
primary_diagnosis_Mucinous adenocarcinoma: 0.0449
primary_diagnosis_Lobular carcinoma, NOS: 0.0443
primary_diagnosis_Metaplastic carcinoma, NOS: 0.0437
primary_diagnosis_Pleomorphic carcinoma: 0.0326
tumor_stage_stage i: 0.032
days_to_death: 0.0222
primary_diagnosis_Infiltrating duct and lobular carcinoma: 0.0166
year_of_death: 0.0154
primary_diagnosis_Infiltrating duct mixed with other types of carcinoma: 0.0154
tumor_stage_stage iib: 0.0148
race_white: 0.0142
tumor_stage_stage ib: 0.0098
tissue_or_organ_of_origin_Breast, NOS: 0.008
```



```
data_21features = data_processed[important_features[:21]]
data_21features.head()
```

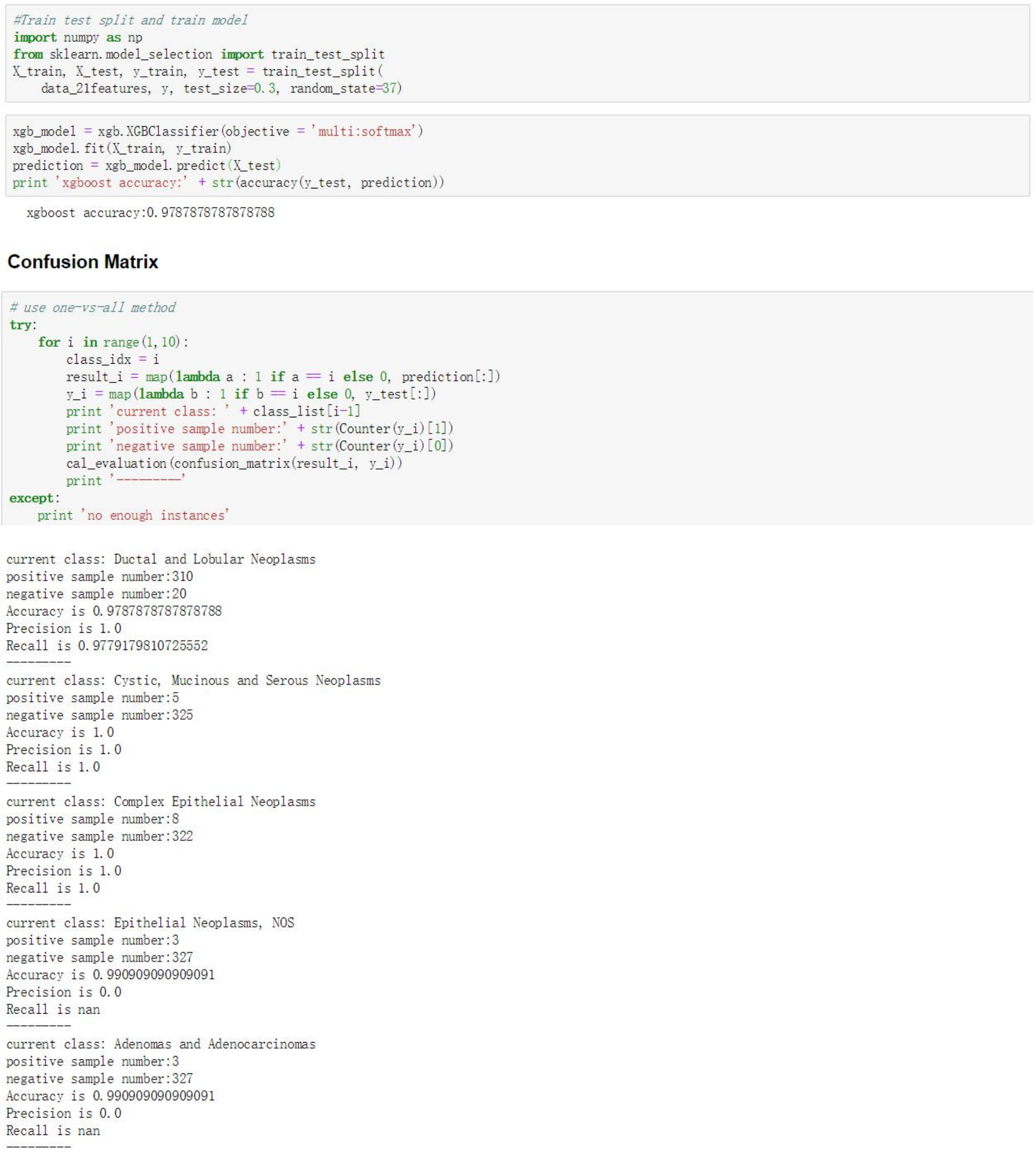
	days_to_last_follow_up	age_at_diagnosis	primary_diagnosis_Infiltrating duct carcinoma, NOS	year_of_birth	race_black or african american	tumor_stage_stage iia	primary_diagnosis_Mucinous adenocarcinoma	primary_
0	1042	19643	1	1955	0	0	0	
1	78	25693	1	1940	0	0	0	
2	624	22849	0	1947	0	1	0	
3	1	20479	0	1957	0	0	0	
4	326	22690	1	1948	0	0	0	

5 rows × 21 columns



```
class_list = ["Ductal and Lobular Neoplasms", "Cystic, Mucinous and Serous Neoplasms", "Complex Epithelial Neoplasms",
              "Epithelial Neoplasms, NOS", "Adenomas and Adenocarcinomas", "Fibroepithelial Neoplasms",
              "Squamous Cell Neoplasms", "Adnexal and Skin Appendage Neoplasms", "Basal Cell Neoplasms"]
xgboost_result_21 = run_cv(X = data_21features, y = y, clf_class= XGBClassifier, objective = 'multi:softmax', num_class = 9)
print 'xgboost accuracy:' + str(accuracy(y, xgboost_result))
for i in range(1,10):
    class_idx = i
    result_i = map(lambda a : 1 if a == i else 0, xgboost_result_21[:])
    y_i = map(lambda b : 1 if b == i else 0, y[:])
    print 'current class: ' + class_list[i-1]
    print 'positive sample number:' + str(Counter(y_i)[1])
    print 'negative sample number:' + str(Counter(y_i)[0])
    cal_evaluation(confusion_matrix(result_i, y_i))
    print '---------'
```

```
current class: Ductal and Lobular Neoplasms
positive sample number:1053
negative sample number:44
Accuracy is 0.9872379216043756
Precision is 1.0
Recall is 0.9868791002811621
---------
current class: Cystic, Mucinous and Serous Neoplasms
positive sample number:16
negative sample number:1081
Accuracy is 1.0
Precision is 1.0
Recall is 1.0
---------
current class: Complex Epithelial Neoplasms
positive sample number:14
negative sample number:1083
Accuracy is 1.0
Precision is 1.0
Recall is 1.0
---------
current class: Epithelial Neoplasms, NOS
positive sample number:5
negative sample number:1092
Accuracy is 0.9954421148587056
Precision is 0.0
Recall is nan
---------
current class: Adenomas and Adenocarcinomas
positive sample number:3
negative sample number:1094
Accuracy is 0.9972652689152234
Precision is 0.0
Recall is nan
---------

current class: Fibroepithelial Neoplasms
positive sample number:2
negative sample number:1095
Accuracy is 0.9981768459434822
Precision is 0.0
Recall is nan
---------
current class: Squamous Cell Neoplasms
positive sample number:2
negative sample number:1095
Accuracy is 0.9981768459434822
Precision is 0.0
Recall is nan
---------
current class: Adnexal and Skin Appendage Neoplasms
positive sample number:1
negative sample number:1096
Accuracy is 0.9990884229717412
Precision is 0.0
Recall is nan
---------
current class: Basal Cell Neoplasms
positive sample number:1
negative sample number:1096
Accuracy is 0.9990884229717412
Precision is 0.0
Recall is nan
---------
```



```python
#Train test split and train model
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    data_21features, y, test_size=0.3, random_state=37)
```

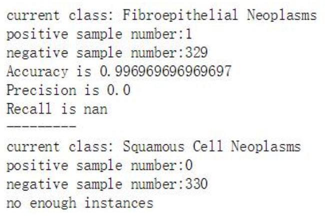
```python
xgb_model = xgb.XGBClassifier(objective = 'multi:softmax')
xgb_model.fit(X_train, y_train)
prediction = xgb_model.predict(X_test)
print 'xgboost accuracy:' + str(accuracy(y_test, prediction))
```

```
xgboost accuracy:0.9787878787878788
```

Confusion Matrix

```python
# use one-vs-all method
try:
    for i in range(1,10):
        class_idx = i
        result_i = map(lambda a : 1 if a == i else 0, prediction[:])
        y_i = map(lambda b : 1 if b == i else 0, y_test[:])
        print 'current class: ' + class_list[i-1]
        print 'positive sample number:' + str(Counter(y_i)[1])
        print 'negative sample number:' + str(Counter(y_i)[0])
        cal_evaluation(confusion_matrix(result_i, y_i))
        print '---------'
except:
    print 'no enough instances'
```

```
current class: Ductal and Lobular Neoplasms
positive sample number:310
negative sample number:20
Accuracy is 0.9787878787878788
Precision is 1.0
Recall is 0.9779179810725552
---------
current class: Cystic, Mucinous and Serous Neoplasms
positive sample number:5
negative sample number:325
Accuracy is 1.0
Precision is 1.0
Recall is 1.0
---------
current class: Complex Epithelial Neoplasms
positive sample number:8
negative sample number:322
Accuracy is 1.0
Precision is 1.0
Recall is 1.0
---------
current class: Epithelial Neoplasms, NOS
positive sample number:3
negative sample number:327
Accuracy is 0.990909090909091
Precision is 0.0
Recall is nan
---------
current class: Adenomas and Adenocarcinomas
positive sample number:3
negative sample number:327
Accuracy is 0.990909090909091
Precision is 0.0
Recall is nan
---------
```



```
current class: Fibroepithelial Neoplasms
positive sample number:1
negative sample number:329
Accuracy is 0.996969696969697
Precision is 0.0
Recall is nan
---------
current class: Squamous Cell Neoplasms
positive sample number:0
negative sample number:330
no enough instances
```