In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
#A1, A4, A5, A6, A8, A9, A11, A12 are categorical attributes.
#A2, A3, A7, A10, A13, A14 are numerical attributes.
#A10, A13, A14 are integers.
#A15 is class attribute: 1 (or '+' at df_org) means approval of the cr
edit, and 0 (or '-') means refusal.

#below is the table as it is published at UCI ML Repository: attribute
names and values are anonymous, and,
#values for categorical attributes are in integer format, which is in
line with the original hierarchy among them
#in such a way that the bigger the number the bigger the likelihood of
approval, within each attribute.
df = pd.read_excel("australian.xlsx")
df_dum  = pd.get_dummies(df)


#and this is the table as it is at its source: attribute names and val
ues are anonymous, and,
# values for categorical attributes are in string format.
df_org = pd.read_excel("australian_org.xlsx")
df_org_dum = pd.get_dummies(df_org)
df_org_dum_model = df_org_dum.iloc[:,:-1]
```

In [3]:
```
df.head()
```

Out[3]:

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
0	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213	0
1	0	22.67	7.00	2	8	4	0.165	0	0	0	0	2	160	1	0
2	0	29.58	1.75	1	4	4	1.250	0	0	0	1	2	280	1	0
3	0	21.67	11.50	1	5	3	0.000	1	1	11	1	2	0	1	1
4	1	20.17	8.17	2	6	4	1.960	1	1	14	0	2	60	159	1

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   A1      690 non-null    int64  
 1   A2      690 non-null    float64
 2   A3      690 non-null    float64
 3   A4      690 non-null    int64  
 4   A5      690 non-null    int64  
 5   A6      690 non-null    int64  
 6   A7      690 non-null    float64
 7   A8      690 non-null    int64  
 8   A9      690 non-null    int64  
 9   A10     690 non-null    int64  
 10  A11     690 non-null    int64  
 11  A12     690 non-null    int64  
 12  A13     690 non-null    int64  
 13  A14     690 non-null    int64  
 14  A15     690 non-null    int64  
dtypes: float64(3), int64(12)
memory usage: 81.0 KB
```

In [5]:
```
df.describe()
```

Out[5]:

	A1	A2	A3	A4	A5	A6	A7	
count	690.000000	690.000000	690.000000	690.000000	690.000000	690.000000	690.000000	6
mean	0.678261	31.568203	4.758725	1.766667	7.372464	4.692754	2.223406	
std	0.467482	11.853273	4.978163	0.430063	3.683265	1.992316	3.346513	
min	0.000000	13.750000	0.000000	1.000000	1.000000	1.000000	0.000000	
25%	0.000000	22.670000	1.000000	2.000000	4.000000	4.000000	0.165000	
50%	1.000000	28.625000	2.750000	2.000000	8.000000	4.000000	1.000000	
75%	1.000000	37.707500	7.207500	2.000000	10.000000	5.000000	2.625000	
max	1.000000	80.250000	28.000000	3.000000	14.000000	9.000000	28.500000	

In [ ]:
```
plt.figure(figsize=(24,24))
sns.pairplot(df)
plt.show()
```

In [ ]:
```
plt.figure(figsize=(24,24))
sns.heatmap(df_org_dum.corr(),linewidths=0.5, annot=True)
plt.show()
```

In [ ]:
```
from sklearn.model_selection import train_test_split, KFold, cross_val
_score
from sklearn.metrics import accuracy_score, confusion_matrix, classifi
cation_report, fbeta_score

from sklearn.model_selection import GridSearchCV

X = df.drop('A15', 1).values
y = df["A15"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.3)

#Naive Bayes methods:
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB

#Support Vector Machines (SVMs):
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
from sklearn.svm import NuSVC

#Nearest Neighbors Classification (NNC):
from sklearn.neighbors import KNeighborsClassifier

#Others:
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

models = [
('GSB' ,  GaussianNB()),
('MNB' ,  MultinomialNB()),
('BNB' ,  BernoulliNB()),
('LSVM' ,  LinearSVC()),
('SVM' ,  SVC()),
('NSVM' ,  NuSVC()),
('KNNC' ,  KNeighborsClassifier()),
('LR', LogisticRegression()),
('DTC', DecisionTreeClassifier()),
('RFC', RandomForestClassifier()),
('LDA', LinearDiscriminantAnalysis())
]
```

```
results = []
names = []
scoring = 'recall'

for name, model in models:
        kfold = KFold(n_splits=20)
        cv_results = cross_val_score(model, X_train, y_train, cv=kfold
, scoring=scoring)
        results.append(cv_results)
        names.append(name)
        nms = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std
())
        print(nms)
```

In [ ]:
```
fig = plt.figure(figsize=(11,5))
fig.suptitle('Comparison of Cross Validation Scores for df')
ax = fig.add_subplot()
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

In [ ]:
```
modelsx = [
('GSB' ,  GaussianNB()),
('MNB' ,  MultinomialNB()),
('BNB' ,  BernoulliNB()),
('LSVM' ,  LinearSVC()),
('SVM' ,  SVC()),
('NSVM' ,  NuSVC()),
('KNNC' ,  KNeighborsClassifier()),
('LR', LogisticRegression()),
('DTC', DecisionTreeClassifier()),
('RFC', RandomForestClassifier()),
('LDA', LinearDiscriminantAnalysis())
]

Xx = df_org_dum_model.drop('A15_+', 1).values
yx = df_org_dum_model["A15_+"].values

Xx_train, Xx_test, yx_train, yx_test = train_test_split(Xx, yx, test_s
ize = 0.3)

resultsx = []
namesx = []
scoringx = 'recall'

for namex, modelx in modelsx:
        kfoldx = KFold(n_splits=20)
        cv_resultsx = cross_val_score(modelx, Xx_train, yx_train, cv=k
foldx, scoring=scoringx)
        resultsx.append(cv_resultsx)
        namesx.append(namex)
        nmsx = "%s: %f (%f)" % (namex, cv_resultsx.mean(), cv_resultsx
.std())
        print(nmsx)
```

In [ ]:
```
fig = plt.figure(figsize=(11,5))
fig.suptitle('Comparison of Cross Validation Scores for df_org_dum_mod
el')
ax = fig.add_subplot()
plt.boxplot(resultsx)
ax.set_xticklabels(namesx)
plt.show()
```

In [ ]:
```
from sklearn import tree
from sklearn.tree import plot_tree


blind_df = df_org_dum_model.iloc[:, :-1]
fn = blind_df.columns

clf = DecisionTreeClassifier()

clf = clf.fit(Xx_train, yx_train)

yx_pred = clf.predict(Xx_test)

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (24,24), dpi=30
0)

tree.plot_tree(clf,
           feature_names = fn,
           class_names=['0','1'],
           filled = True,
           rotate = True,
           proportion = True,
           precision=1);

fig.savefig('CreditApproval.png')
```

In [ ]:
```
import statsmodels.api as sm

yxx = df_org_dum_model.iloc[:,-1]
Xxx = df_org_dum_model.iloc[:,:-1]

logit_model = sm.Logit(yxx,Xxx)
resultxx = logit_model.fit()

print(resultxx.summary())
```

In [ ]:
```
import statsmodels.api as sm

yxxx = df.iloc[:,-1]
Xxxx = df.iloc[:,:-1]

logit_model = sm.Logit(yxxx,Xxxx)
resultxxx = logit_model.fit()

print(resultxxx.summary())
```