In [155]:
```
import pandas as pd
import numpy as np
import sklearn as sk
from sklearn import linear_model
from sklearn import svm
from sklearn import tree
from sklearn import naive_bayes
from sklearn import neighbors
from sklearn import naive_bayes
import matplotlib.pyplot as plt
```

Data used: Adult Data Set - Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset. Link to the data: https://archive.ics.uci.edu/ml/datasets/Adult (https://archive.ics.uci.edu/ml/datasets/Adult)

The data came in a CSV file; before loading the data into jupyter, I have made few data transformation by hand - for categories sex, race, relationship, marital-status, workclass, education, ocupation, native country and income_group, I have transformed the data from text to numerical data. Basically assigning to each type within the given category (eg. for sex, 0 was Male, 1 was Female). The table below shows an overview of assigned values per category.

In [273]:
```
# Here dataset is loaded to jupyter
data = pd.read_excel('C:/Users/janvr/Desktop/coursera/data analysis with pyt
hon/course 2 - design thinking and predictive DA/week5/adult_.xlsx', sheet_n
ame='Adjusted')
```

In [157]:
```
# Here the legend is loaded - showing an overview of assigned values (column [value]) per category
legend = pd.read_excel('C:/Users/janvr/Desktop/coursera/data analysis with python/course 2 - design thinking and predictive DA/week5/adult_.xlsx', sheet_name='legend')
legend
```

Out[157]:

	value	sex	race	relationship	marital-status	workclass	education	occupation-num	native-country-num
0	0	Male	White	Other-relative	Married-civ-spouse	Self-emp-not-inc	NaN	Exec-managerial	United-States
1	1	Female	Black	Unmarried	Divorced	Private	Preschool	Handlers-cleaners	Cuba
2	2	NaN	Asian-Pac-Islander	Husband	Married-spouse-absent	State-gov	1st-4th	Prof-specialty	Jamaica
3	3	NaN	Amer-Indian-Eskimo	Not-in-family	Never-married	Federal-gov	5th-6th	Other-service	India
4	4	NaN	Other	Own-child	Separated	Local-gov	7th-8th	Adm-clerical	?
5	5	NaN	NaN	Wife	Married-AF-spouse	?	9th	Sales	Mexico
6	6	NaN	NaN	NaN	Widowed	Self-emp-inc	10th	Craft-repair	South
7	7	NaN	NaN	NaN	NaN	Without-pay	11th	Transport-moving	Puerto-Rico
8	8	NaN	NaN	NaN	NaN	Never-worked	12th	Farming-fishing	Honduras
9	9	NaN	NaN	NaN	NaN	NaN	HS-grad	Machine-op-inspct	England
10	10	NaN	NaN	NaN	NaN	NaN	Some-college	Tech-support	Canada
11	11	NaN	NaN	NaN	NaN	NaN	Assoc-voc	?	Germany
12	12	NaN	NaN	NaN	NaN	NaN	Assoc-acdm	Protective-serv	Iran
13	13	NaN	NaN	NaN	NaN	NaN	Bachelors	Armed-Forces	Philippines
14	14	NaN	NaN	NaN	NaN	NaN	Masters	Priv-house-serv	Italy
15	15	NaN	NaN	NaN	NaN	NaN	Prof-school	NaN	Poland
16	16	NaN	NaN	NaN	NaN	NaN	Doctorate	NaN	Columbia
17	17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Cambodia
18	18	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Thailand
19	19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Ecuador
20	20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Laos
21	21	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Taiwan
22	22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Haiti
23	23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Portugal
24	24	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Dominican-Republic
25	25	NaN	NaN	NaN	NaN	NaN	NaN	NaN	El-Salvador
26	26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	France
27	27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Guatemala
28	28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	China
29	29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Japan

In [296]:
```
# here i am just showing the dataseet for illustration
data
```

Out[296]:

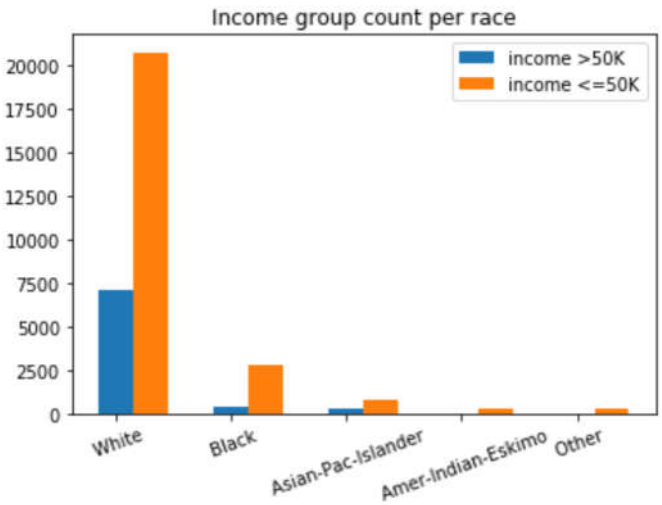
	age	workclass	fnlwgt	education-num	marital-status	occupation-num	relationship	race	sex	capital-gain	capi l
0	39	2	77516	13	3	4	3	0	0	2174	
1	50	0	83311	13	0	0	2	0	0	0	
2	38	1	215646	9	1	1	3	0	0	0	
3	53	1	234721	7	0	1	2	1	0	0	
4	28	1	338409	13	0	2	5	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	
32556	27	1	257302	12	0	10	5	0	1	0	
32557	40	1	154374	9	0	9	2	0	0	0	
32558	58	1	151910	9	6	4	1	0	1	0	
32559	22	1	201490	9	3	4	4	0	0	0	
32560	52	6	287927	9	0	0	5	0	1	15024	

32561 rows × 15 columns

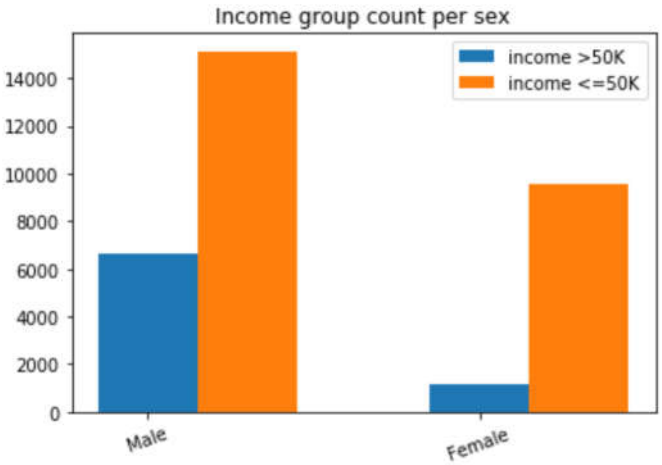
There were no outliers, no n/a in the dataset itself.

In the. following section I am adding some graphs to give an illustration of the data.

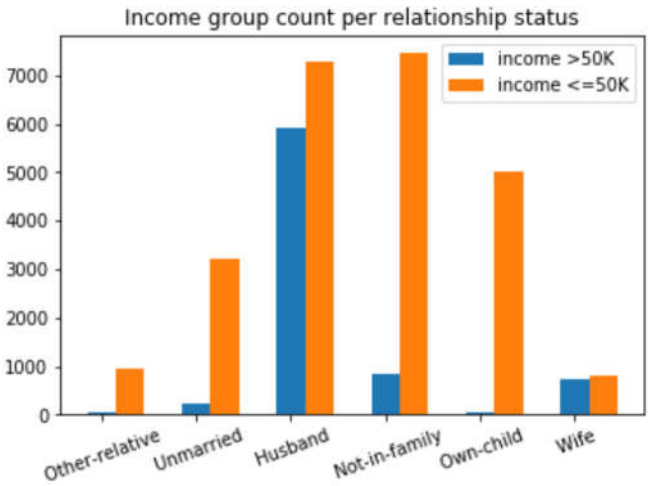
In [233]:
```
#Graph of number of people per race and per income group. We see that white
people are disproportionally
#more often income over >50K.

d2=pd.DataFrame()
d2['race'], d2['income_group']=data['race'], data['income_group']
g_race=d2[d2['income_group']==1].groupby(['race']).count()
g_race_=d2[d2['income_group']==0].groupby(['race']).count()
plt.bar(g_race.index,g_race['income_group'], width=0.3, label='income >50K')
plt.bar(g_race.index+0.3 ,g_race_['income_group'], width=0.3, label='income
<=50K')
plt.xticks(g_race.index,legend['race'].dropna(), rotation=20)
plt.title('Income group count per race')
plt.legend()
plt.show()
```



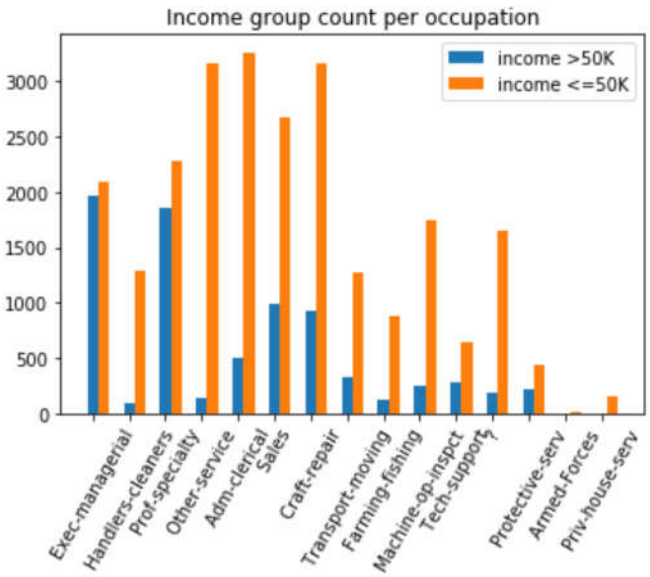
In [234]:
```
#Income group per sex. Men have approx 6times more often income above 50K th
an women.

d2=pd.DataFrame()
d2['sex'], d2['income_group']=data['sex'], data['income_group']
g_race=d2[d2['income_group']==1].groupby(['sex']).count()
g_race_=d2[d2['income_group']==0].groupby(['sex']).count()
plt.bar(g_race.index,g_race['income_group'], width=0.3, label='income >50K')
plt.bar(g_race.index+0.3 ,g_race_['income_group'], width=0.3, label='income
<=50K')
plt.xticks(g_race.index,legend['sex'].dropna(), rotation=20)
plt.title('Income group count per sex')
plt.legend()
plt.show()
```



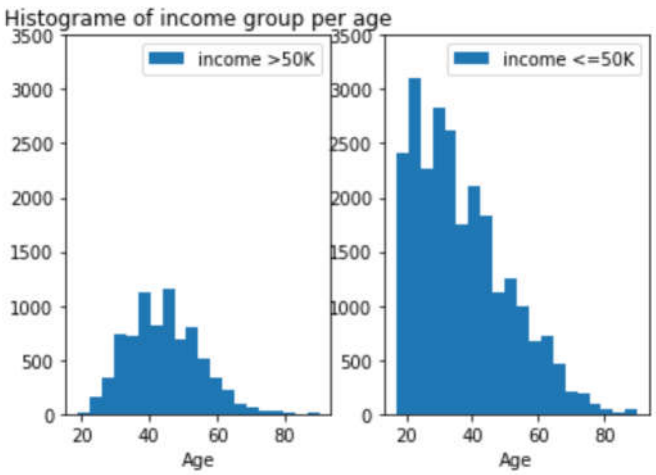
In [235]:
```
#Number of people per income group and per relationship status.
#Husbands have are by far the ones have income >50K.

d2=pd.DataFrame()
d2['relationship'], d2['income_group']=data['relationship'], data['income_gr
oup']
g_race=d2[d2['income_group']==1].groupby(['relationship']).count()
g_race_=d2[d2['income_group']==0].groupby(['relationship']).count()
plt.bar(g_race.index,g_race['income_group'], width=0.3, label='income >50K')
plt.bar(g_race.index+0.3 ,g_race_['income_group'], width=0.3, label='income
<=50K')
plt.xticks(g_race.index,legend['relationship'].dropna(), rotation=20)
plt.title('Income group count per relationship status')
plt.legend()
plt.show()
```



In [297]:
```
#Number of people per income group and type of occuppation.
#Here the legend has moved and I did not manage to fix it, however executive
managers and professional
#specialists have income >50K the most.

d2=pd.DataFrame()
d2['occupation-num'], d2['income_group']=data['occupation-num'], data['incom
e_group']
g_race=d2[d2['income_group']==1].groupby(['occupation-num']).count()
g_race_=d2[d2['income_group']==0].groupby(['occupation-num']).count()
plt.bar(g_race.index,g_race['income_group'], width=0.3, label='income >50K')
plt.bar(g_race.index+0.3 ,g_race_['income_group'], width=0.3, label='income
<=50K')
plt.xticks(g_race.index,legend['occupation-num'].dropna(), rotation=60)
plt.title('Income group count per occupation')
plt.legend()
plt.show()
```



In [269]:
```
#Histograms of number of people per income group per age.
# People between approx 35 and 50 are the ones having income >50K the most o
ften.

d2=pd.DataFrame()
d2['age'], d2['income_group']=data['age'], data['income_group']
g_race=d2[d2['income_group']==1]
g_race_=d2[d2['income_group']==0]

plt.subplot(121)
plt.title('Histograme of income group per age')
plt.hist(g_race['age'], bins=20,label='income >50K')
plt.ylim([0,3500])
plt.legend()
plt.xlabel('Age')
plt.subplot(122)
plt.hist(g_race_['age'], bins=20, label='income <=50K')
plt.ylim([0,3500])
plt.legend()
plt.xlabel('Age')
plt.show()
```



In [285]:
```
#Here I divided the data to X (all characteristics and y(income group) and a
lso created a train and
#test group

X=data.iloc[:,:-2]
y=data.iloc[:,-2]
X_train=X[:(len(X)//2)]
X_test=X[(len(X)//2):]
y_train=y[:(len(X)//2)]
y_test=y[(len(X)//2):]
```

In [286]:
```
#Logistic regression model -> gives 80% accuracy of predictions. As we will
see later, this is one of the
#best results.

model1=linear_model.LogisticRegression()
model1.fit(X_train, y_train)
predictions1=model1.predict(X_test)
correctpredictions1 = predictions1 == y_test
s1=sum(correctpredictions1)/len(correctpredictions1)
s1
```

```
C:\Users\janvr\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.p
y:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Sp
ecify a solver to silence this warning.
  FutureWarning)
```

Out[286]:
```
0.800135126834961
```

In [287]:
```
#Linear regression, accuracy is 0%. This is not surprising given that we are
dealing with binary features
#for which linear model is not suitable at all.

model2=linear_model.LinearRegression()
model2.fit(X_train, y_train)
predictions2=model2.predict(X_test)
correctpredictions2 = predictions2 == y_test
s2=sum(correctpredictions2)/len(correctpredictions2)
s2
```

Out[287]:
```
0.0
```

In [288]:
```
#Support vector classifier. Prediction accuracy is 76%. I believe this could
be further improved by
#transforming non-binary characteristics such as age or capital gain to its
logarithmic values.
#that is something I did not do as part of this exercise.

model3=svm.SVC()
model3.fit(X_train, y_train)
predictions3=model3.predict(X_test)
correctpredictions3 = predictions3 == y_test
s3=sum(correctpredictions3)/len(correctpredictions3)
s3
```

```
C:\Users\janvr\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureW
arning: The default value of gamma will change from 'auto' to 'scale' in ve
rsion 0.22 to account better for unscaled features. Set gamma explicitly to
'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[288]:
```
0.7579387015539586
```

In [289]:
```
#Decision tree classifier. Prediction accuracy is 81% which is not bad.

model4=tree.DecisionTreeClassifier()
model4.fit(X_train, y_train)
predictions4=model4.predict(X_test)
correctpredictions4 = predictions4 == y_test
s4=sum(correctpredictions4)/len(correctpredictions4)
s4
```

Out[289]:
```
0.8129107548676371
```

In [292]:
```
#Nearest neighbour classifies - prediction accuracy 77%

model5=neighbors.KNeighborsClassifier()
model5.fit(X_train, y_train)
predictions5=model5.predict(X_test)
correctpredictions5 = predictions5 == y_test
s5=sum(correctpredictions5)/len(correctpredictions5)
s5
```

Out[292]: 0.7715742276272957

In [293]:
```
#Naive Bayes - prediction accuracy 78%.

model6=naive_bayes.MultinomialNB()
model6.fit(X_train, y_train)
predictions6=model6.predict(X_test)
correctpredictions6 = predictions6 == y_test
s6=sum(correctpredictions6)/len(correctpredictions6)
s6
```

Out[293]: 0.7831828511762177

Conclusion: Decision Tree classifier and Logistic regression are the ones with the most prediction accuracy.