Data Analytics Lab: Assignment - 3
A Mathematical Essay on Naive Bayes Classifier

Nagappan N
Department of Metallurgical and Materials Engineering
IIT Madras
Chennai, India
mm19b040@smail.iitm.ac.in

***Abstract*—The aim of this project is to predict whether or not a person earns more than $50k per year. We have used the Bernoulli Naive Bayes classifier to achieve this. Our model is able to predict with an accuracy of 79.85%.**

I. Introduction

This project mainly aims to predict whether or not a person earns more than $50k per year. We try to see what are the factors that results in more earnings for a person. We use the 1994 Census bureau database by Ronny Kohavi and Barry Becker for this purpose.

The ”naive” assumption that each pair of features is conditionally independent given the value of the class variable underlies a class of supervised learning algorithms collectively referred to as ”naive Bayes methods.” Bernoulli Naive Bayes is used for data that is distributed according to multivariate Bernoulli distributions.

Using the naive bayes classifier, we predict whether or not a person earns more than $50k per year. We use the input features of age, person's current job related information, educational status, marital status, sex, race and native country for this prediction. The model developed would help us in making some crucial inferences. A similar version of this model can be used a way to classify below above and below the poverty line. Hence, this kind of model would be very handy for government based agencies and NGOs.

This paper starts with the description of naive bayes classifier. It is followed by the description of the datasets. This includes data visualization and processing. The following section includes details about how the model has been implemented. Finally, we conclude with the key inferences from our project.

II. Naive Bayes Classifier

A group of supervised learning algorithms known as naive Bayes methods utilise Bayes' theorem with the ”naive” assumption that each pair of features is conditionally independent given the value of the class variable. Given the class variable y and the dependent feature vectors x_1 through X_n, the Bayes theorem describes the relationship in the following way.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

Then we may estimate P(y) and P(x_i—y) using Maximum A Posteriori (MAP) estimation; the former is then the relative frequency of class y in the training set.

The assumptions that different naive Bayes classifiers make about the distribution of P(x_i—y) are what distinguish them from one another:

- Gaussian Naive Bayes: GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian.
- Multinomial Naive Bayes: MultinomialNB implements the naive Bayes algorithm for multinomially distributed data, The parameters y is estimated by a smoothed version of maximum likelihood.
- Complement Naive Bayes: CNB is an adaptation of the standard multinomial naive Bayes algorithm for imbalanced data sets. CNB uses statistics from the complement of each class to compute the model's weights.
- Categorical Naive Bayes: CategoricalNB implements the categorical naive Bayes algorithm for categorically distributed data. It assumes that each feature, which is described by the index i, has its own categorical distribution.
- Bernoulli Naive Bayes: BernoulliNB is used for data that is distributed according to multivariate Bernoulli distributions; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. The expression for this is as follows:

$$P(x_i \mid y) = P(x_i = 1 \mid y) x_i + (1 - P(x_i = 1 \mid y))(1 - x_i)$$

III. Datasets

The dataset has the following columns:

1) age: The age of the person. It is a continuous variable.
2) work class: This is a categorical variable describing the working class of the person. It takes 8 values: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3) fnlwgt:
4) education: The level of education obtained by the person. This is again a categorical variable which takes one of the 16 values: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5) education-num: Number of years of education pursued by the person.
6) marital status: The marital status of the person is also a categorical variable which takes one of the following values: marital-status Marital status Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7) Occupation: This is also a categorical variable describing the occupation of the person. It takes one of these values: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
8) relationship: This is defined as a categorical variable that can take the values: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9) race: This categorical variables describes the race of the person which can be White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other or Black.
10) sex: The sex of the person.
11) capital gain
12) capital loss
13) Hours-per-week: THe number of hours a person works in a week is given by this variable.
14) Native-country: The country that the person belongs to.
15) salary: Tells us if the person earns greater than \$50k or lesser than \$50k per year.

A. Data Processing

First, we visualize the missing values in the dataset.

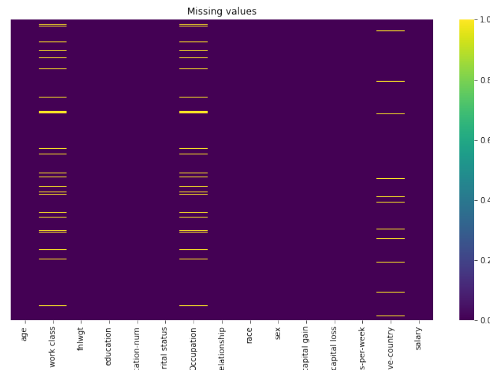


Fig. 1. Heatmap showing the distribution of missing values in our dataset

The employment variables work-class, occupation, and native-country all have missing values. We decide to add a distinct category for missing data as "Unknown" because the percentage of missing values is larger.

We can now clean the data and discretize the continuous variables of age, hours-per-week, education count, capital loss, and capital gain into categorical ones based on the relative density distributions of their salary class since we have the highest number of input variables of the categorical type. In order to condense the enormous number of classes, we also examine each categorical variable's relative density distribution for its respective salary class.

- Work class: Here, we consolidate all federal, local, and state government-related classes. We also eliminate redundancy by integrating the never-worked and unpaid job classes. Additionally, as the pay scales for the self-employed and the private sector are comparable, we have combined the two.

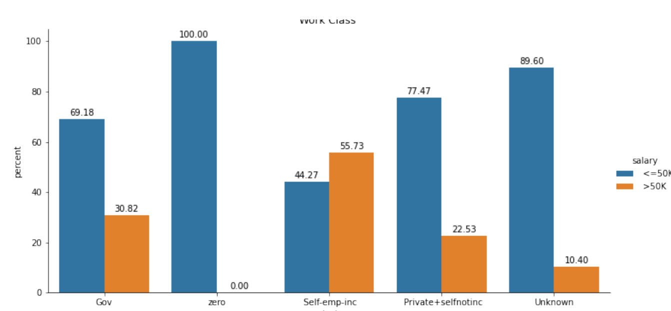


Fig. 2. Percentage of people who earn above and below \$50k per year based on their work class.

- Number of years of education: We can see a clear difference after 13 years of education in the salary fractions this is a direct correlation between persons who completed schooling and people who did not.

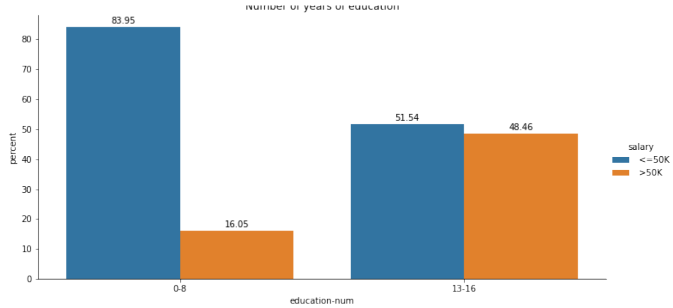


Fig. 3. Percentage of people who earn above and below $50k per year based on number of years of education obtained by the person.

- Education: In this column we can combine whole of schooling together and also combine Bachelor and master degree holders. We can also see that doctoral and professional school education levels are comparable, thus we can combine the two. HS graduates and other college groups are also able to be combined. Additionally, an Assoc category may combine the Assoc-voc and Assoc-acdm categories.

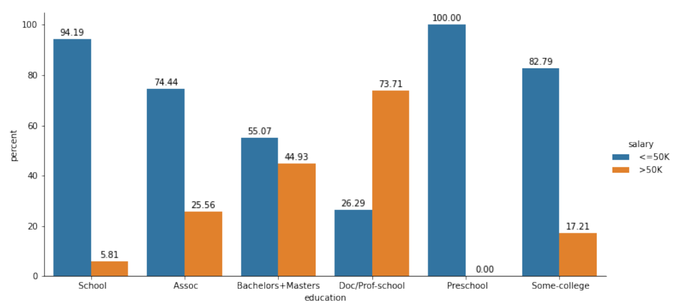


Fig. 4. Percentage of people who earn above and below $50k per year based on their education.

- Race: We see white and Asians have higher salary groups than other races.

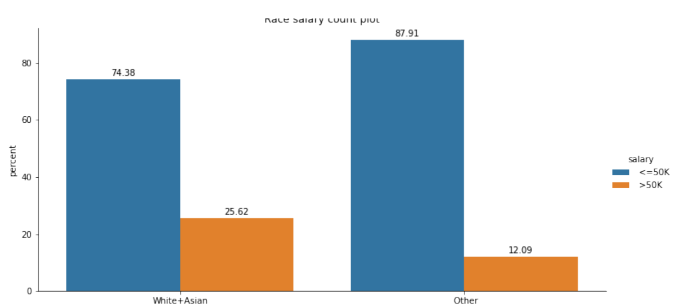


Fig. 5. Percentage of people who earn above and below $50k per year based on the person's race.

- Sex: We see that males have higher salary fraction than females

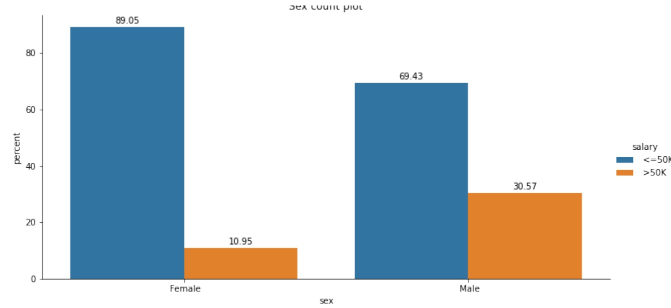


Fig. 6. Percentage of people who earn above and below $50k per year based on the person's sex.

- Age: Based on the following plot we discretize age into three categories in intervals of 40 and as the fraction of salaries are similar in the first and third categories we combine them.

Fig. 7. Percentage of people who earn above and below $50k per year based on the person's age.

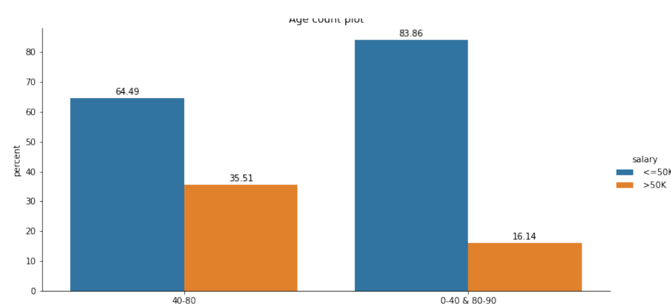


Fig. 8. Histogram plot using polynomial element for the variable age.

- Hours per week count plot: We can see from the probability graph that we can discretize the domain into 0-50 and 50-100.

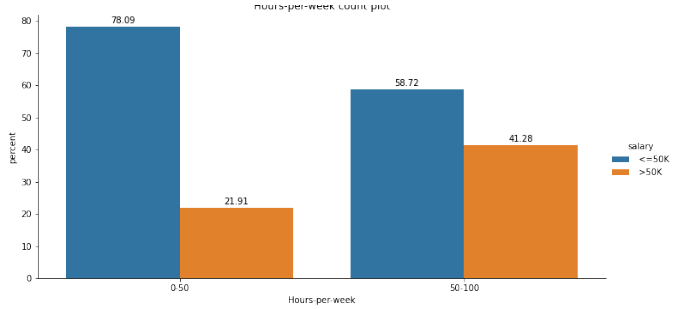


Fig. 9. Histogram plot using polynomial element for the variable hours per week.

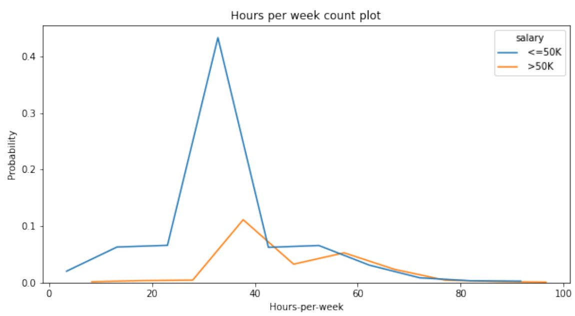


Fig. 10. Percentage of people who earn above and below $50k per year based on number of hours the person works.

- Capital gain and capital loss: Using probability density charts, we discretize and combine the comparable categories in a manner similar to the aforementioned continuous plots.

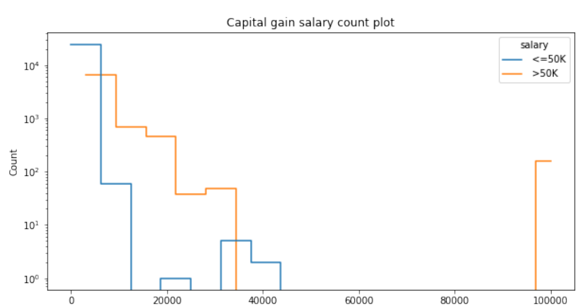


Fig. 11. Histogram plot using step element for the variable capital gain.

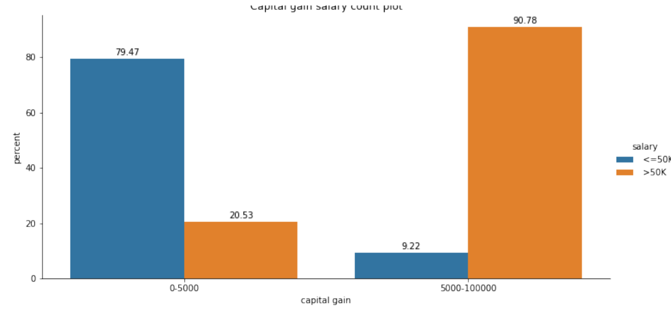


Fig. 12. Percentage of people who earn above and below $50k per year based on capital gain.

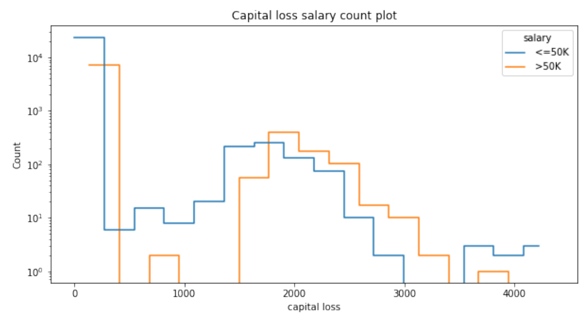


Fig. 13. Histogram plot using step element for the variable capital loss.

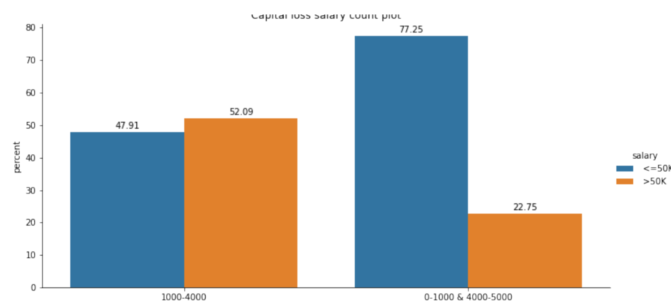


Fig. 14. Percentage of people who earn above and below $50k per year based on capital loss.

- Marital status: We can cut down different number of categories into just two With / without spouse.

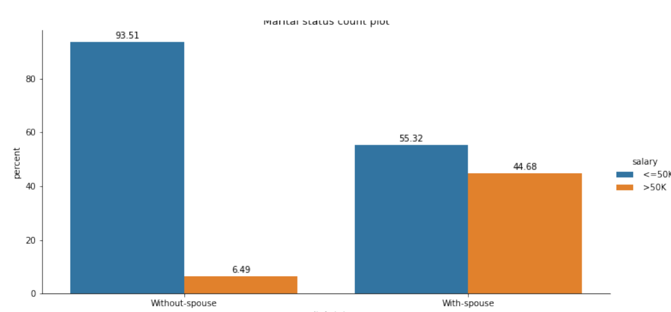


Fig. 15. Percentage of people who earn above and below $50k per year based on marital status.

- Occupation: Based on the wage distributions, we can divide many professions into two main groups: high and low income groups.

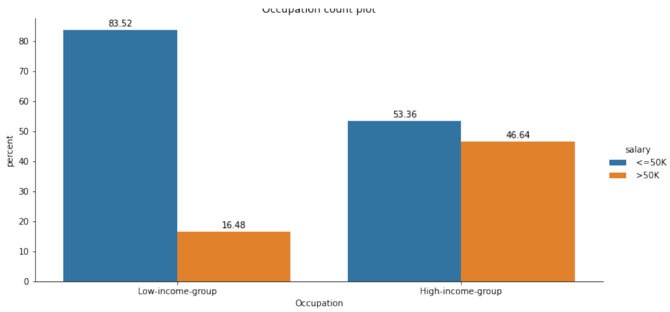


Fig. 16. Percentage of people who earn above and below $50k per year based on occupation.

- Relationship: Here again each category can be brought into whether or not they are in a family.

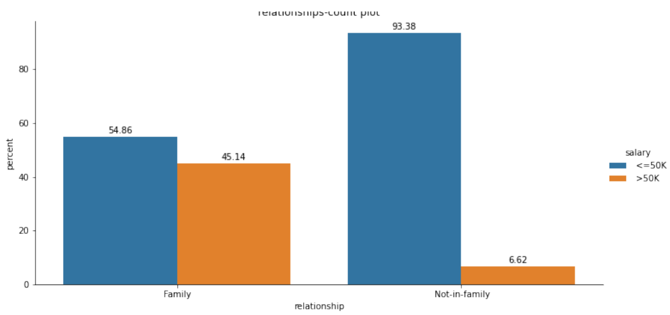


Fig. 17. Percentage of people who earn above and below $50k per year based on relationship.

- Native Country: We can categorise into US and non-US countries because the majority of data is in the US.

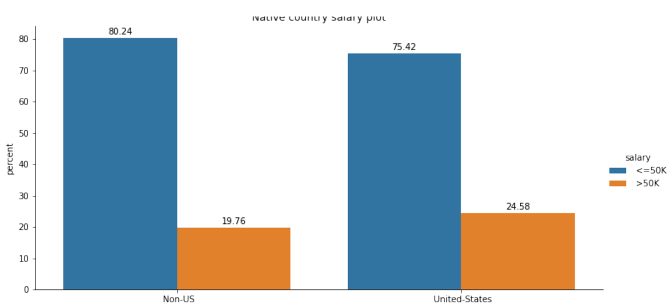


Fig. 18. Percentage of people who earn above and below $50k per year based on native country.

IV. Modelling

We only use columns that, after cleaning, may be divided into only two groups. We drop the work class and education columns as they essentially cover the same information as the occupation and education number columns. So, now, a Bernoulli Naive Bayes classifier model is appropriate.

Before using BernoulliNB to train and forecast the data, we must first use Label Encoder to assign a 0 or 1 to each category. To assess performance following training, we separated the data into train and test datasets.

We may compare the target salary variable's predicted and actual values by utilising the accuracy score as the performance metric. Our model is able to make predictions with an accuracy of 79.85%

V. Conclusions

We find that a simple model like naive bayes is good to make predictions for real world problems like this. We have implemented a Bernoulli Naive Bayes classifier on the dataset to predict whether or not a person earns more than $50k per year with an accuracy of 79.85%.

EE4708: Data Analytics Lab

Assignment 3

Name: Nagappan N

Roll number: MM19B040

Importing libraries

In [75]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

Understanding the Dataset

In [76]:

```
df=pd.read_excel('adult.xlsx')
```

In [77]:

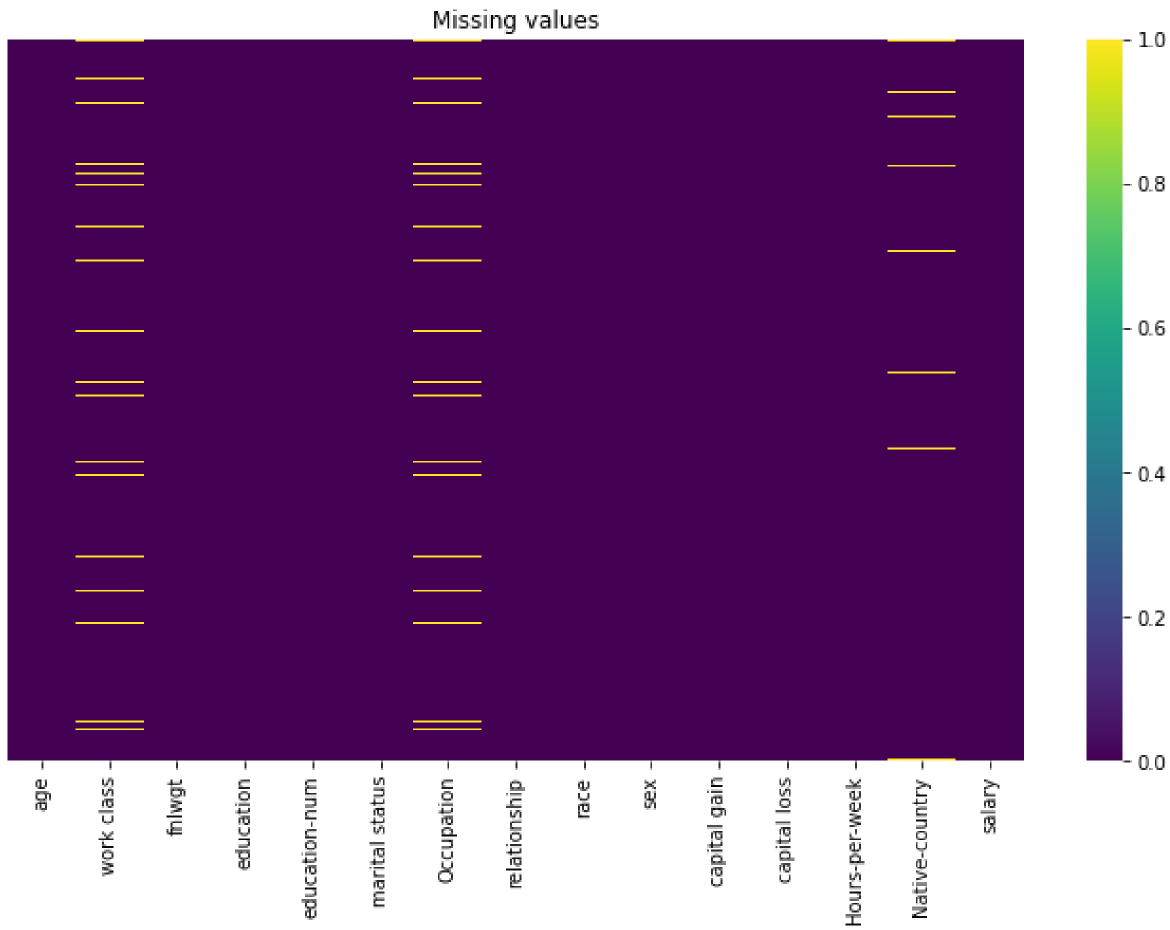
```
df.head()
```

Out[77]:

	age	work class	fnlwgt	education	education-num	marital status	Occupation	relationship	race	se
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Ma
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Ma
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Ma
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fema

In [78]:

```
df.replace('\?',np.nan,regex=True,inplace=True)
plt.figure(figsize=[12,7])
sns.heatmap(df.isnull(), cmap='viridis')
plt.yticks(ticks=[])
plt.title('Missing values')
plt.savefig('Missingvalues.png')
```



As missing values are not easily determinable we create a new category unknown for all missing values alike

In [79]:

```
df.fillna('Unknown',inplace=True)
```

In [80]:

```
df['work class']=df['work class'].astype('category')
df['education']=df['education'].astype('category')
df['marital status']=df['marital status'].astype('category')
df['Occupation']=df['Occupation'].astype('category')
df['relationship']=df['relationship'].astype('category')
df['race']=df['race'].astype('category')
df['sex']=df['sex'].astype('category')
df['Native-country']=df['Native-country'].astype('category')
df['salary']=df['salary'].astype('category')
```

In [81]:

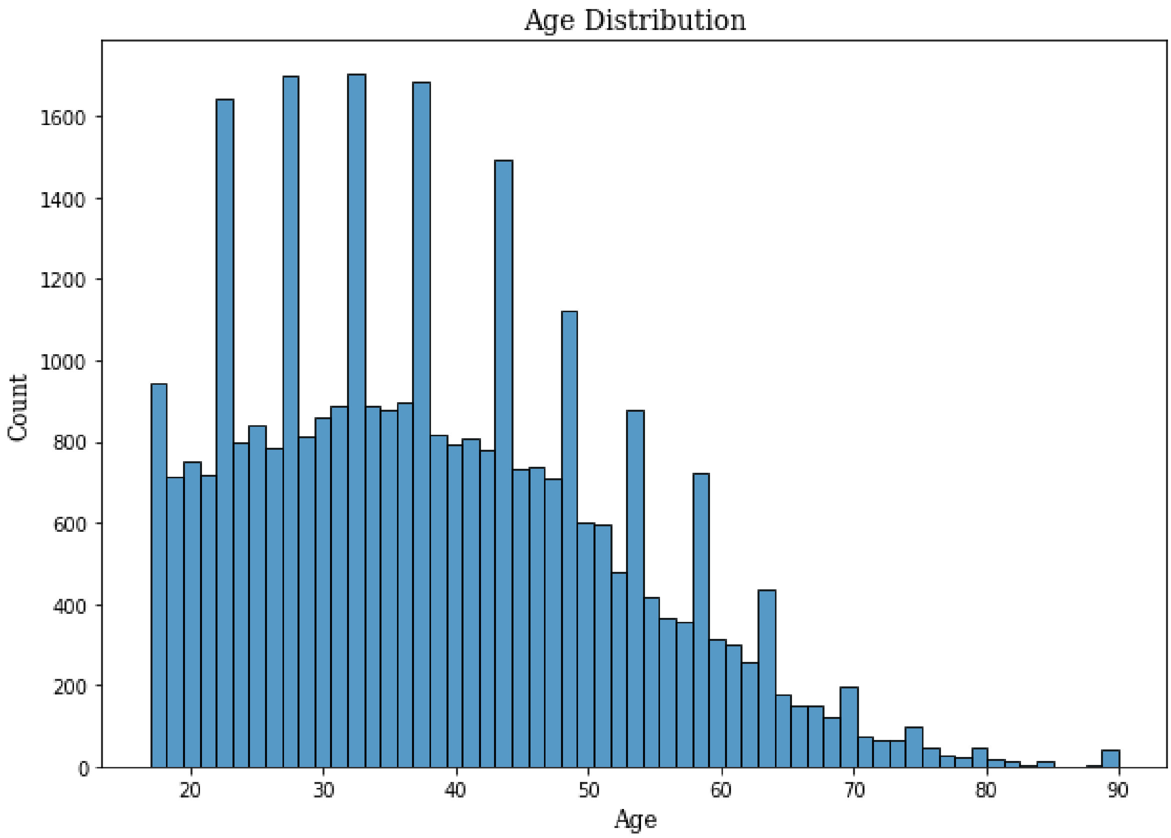
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype   
---  ------          --------------  -----   
 0   age             32561 non-null  int64   
 1   work class      32561 non-null  category
 2   fnlwgt          32561 non-null  int64   
 3   education       32561 non-null  category
 4   education-num   32561 non-null  int64   
 5   marital status  32561 non-null  category
 6   Occupation      32561 non-null  category
 7   relationship    32561 non-null  category
 8   race            32561 non-null  category
 9   sex             32561 non-null  category
 10  capital gain    32561 non-null  int64   
 11  capital loss    32561 non-null  int64   
 12  Hours-per-week  32561 non-null  int64   
 13  Native-country  32561 non-null  category
 14  salary          32561 non-null  category
dtypes: category(9), int64(6)
memory usage: 1.8 MB
```

We next look into how the distribution of few of the continuous features in our dataset.

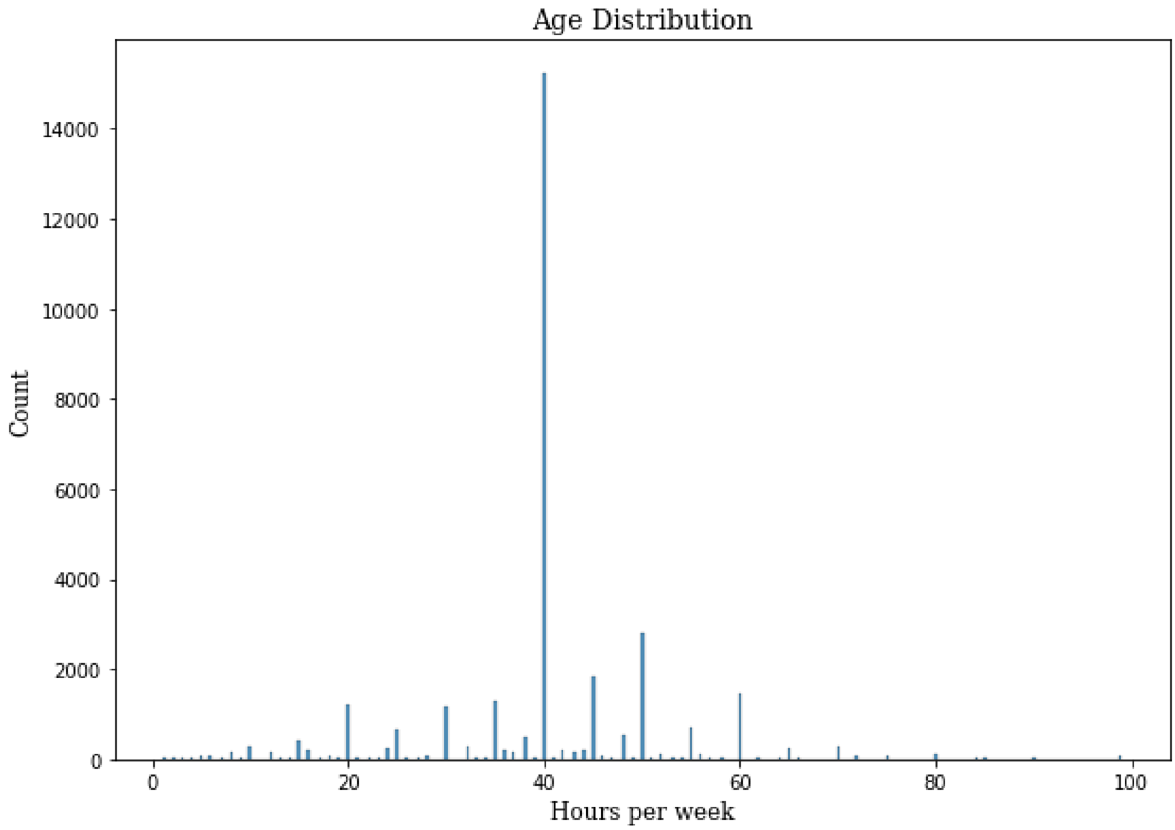
In [82]:

```
plt.figure(figsize=(10,7))
sns.histplot(df['age'])
font2 = {'family':'serif','color':'black','size':12}
font1 = {'family':'serif','color':'black','size':14}
plt.title('Age Distribution',fontdict=font1)
plt.ylabel('Count',fontdict=font2)
plt.xlabel('Age',fontdict=font2)
plt.savefig('agedist.png',dpi=2048)
```



In [83]:

```
plt.figure(figsize=(10,7))
sns.histplot(df['Hours-per-week'])
font2 = {'family':'serif','color':'black','size':12}
font1 = {'family':'serif','color':'black','size':14}
plt.title('Age Distribution',fontdict=font1)
plt.ylabel('Count',fontdict=font2)
plt.xlabel('Hours per week',fontdict=font2)
plt.savefig('hoursdist.png',dpi=2048)
```



As we have maximum categorical variables we need to bring all data in categorical form

Now we proceed to clean and extract the features from the data by following steps

- If it is a continous variable we discretize and then combine discretized categoies of similar income groups to have not more than 2 categories
- If it is a categorical variable we clean it to the best possible extent and use the ones which has only two categories into a bernoulli Naive bayes algorithm.

In [84]:

```
df.groupby('work class')['salary'].value_counts(normalize=True)
```

Out[84]:

```
work class         salary
 Federal-gov        <=50K    0.613542
                    >50K     0.386458
 Local-gov          <=50K    0.705208
                    >50K     0.294792
 Never-worked       <=50K    1.000000
                    >50K     0.000000
 Private            <=50K    0.781327
                    >50K     0.218673
 Self-emp-inc       >50K     0.557348
                    <=50K    0.442652
 Self-emp-not-inc   <=50K    0.715073
                    >50K     0.284927
 State-gov          <=50K    0.728043
                    >50K     0.271957
 Without-pay        <=50K    1.000000
                    >50K     0.000000
Unknown             <=50K    0.895969
                    >50K     0.104031
Name: salary, dtype: float64
```

Lets map all government jobs together and combine private and self-emp-not-inc with private, and also never worked and without pay together to create better categories

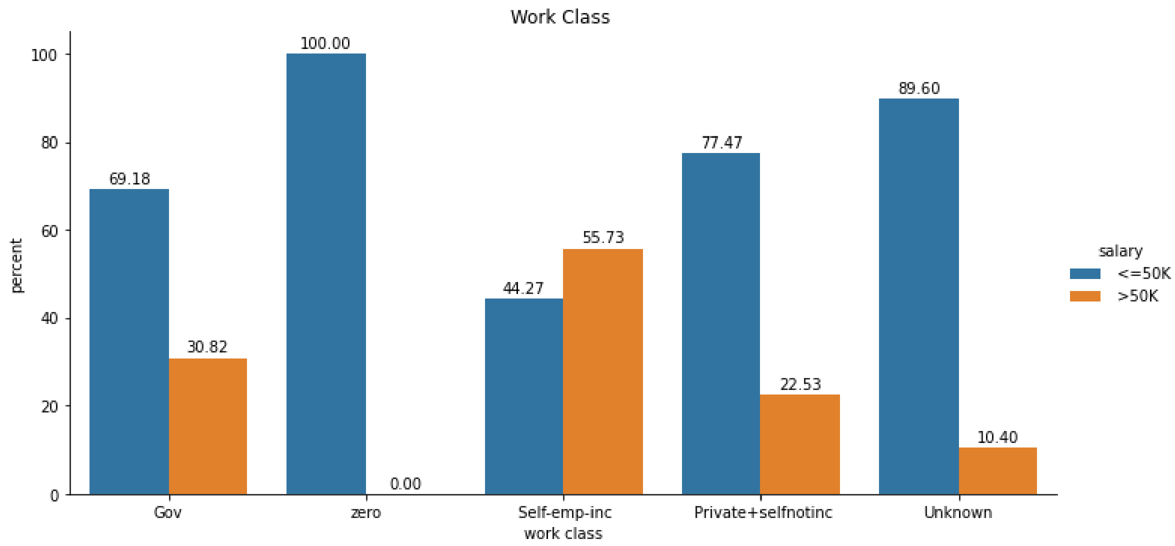
In [85]:

```
mappings={' Federal-gov':'Gov',' Local-gov':'Gov',' State-gov':'Gov',' Never-worked':'zero'
df.replace(mappings,inplace=True)
```

In [86]:

```
x='work class'
y='salary'
dfp=(df
.groupby(x)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x='work class',y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Work Class ')
plt.savefig('workclasscplot2.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [87]:

```
df.groupby('education-num')['salary'].value_counts(normalize=True)
```

Out[87]:

```
education-num  salary
1              <=50K     1.000000
               >50K      0.000000
2              <=50K     0.964286
               >50K      0.035714
3              <=50K     0.951952
               >50K      0.048048
4              <=50K     0.938080
               >50K      0.061920
5              <=50K     0.947471
               >50K      0.052529
6              <=50K     0.933548
               >50K      0.066452
7              <=50K     0.948936
               >50K      0.051064
8              <=50K     0.923788
               >50K      0.076212
9              <=50K     0.840491
               >50K      0.159509
10             <=50K     0.809765
               >50K      0.190235
11             <=50K     0.738784
               >50K      0.261216
12             <=50K     0.751640
               >50K      0.248360
13             <=50K     0.585247
               >50K      0.414753
14             >50K      0.556587
               <=50K     0.443413
15             >50K      0.734375
               <=50K     0.265625
16             >50K      0.740920
               <=50K     0.259080
Name: salary, dtype: float64
```

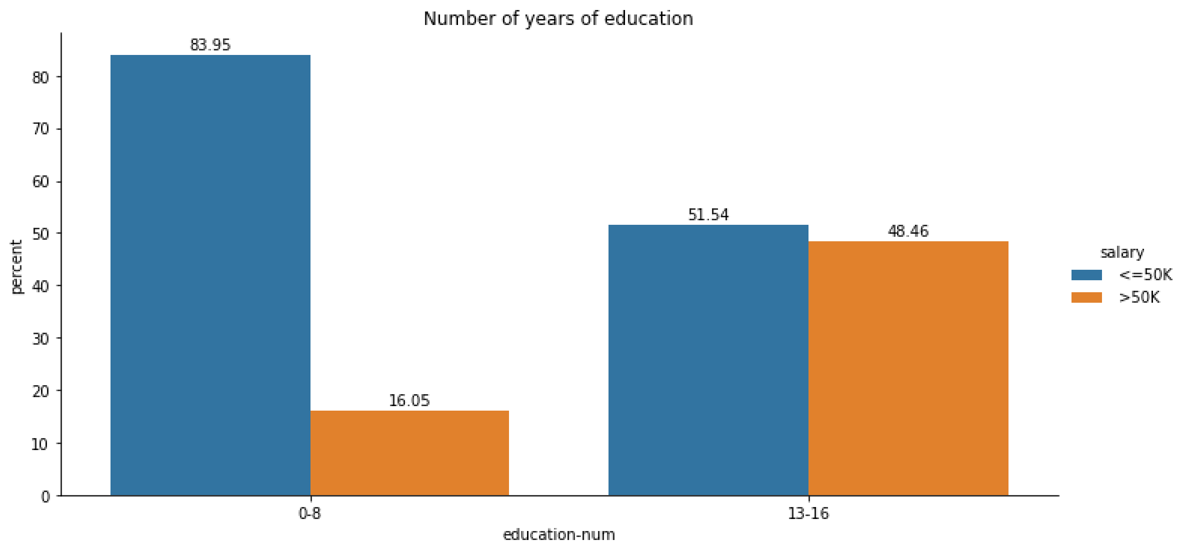
In [88]:

```
df['education-num']=pd.cut(df['education-num'],bins=[0,12,16],labels=['0-8','13-16'])
```

In [89]:

```
x1='education-num'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Number of years of education')
plt.savefig('edunumcplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [90]:

```
df.groupby('education')['salary'].value_counts(normalize=True)
```

Out[90]:

```
education      salary
 10th           <=50K    0.933548
                >50K     0.066452
 11th           <=50K    0.948936
                >50K     0.051064
 12th           <=50K    0.923788
                >50K     0.076212
 1st-4th        <=50K    0.964286
                >50K     0.035714
 5th-6th        <=50K    0.951952
                >50K     0.048048
 7th-8th        <=50K    0.938080
                >50K     0.061920
 9th            <=50K    0.947471
                >50K     0.052529
 Assoc-acdm     <=50K    0.751640
                >50K     0.248360
 Assoc-voc      <=50K    0.738784
                >50K     0.261216
 Bachelors      <=50K    0.585247
                >50K     0.414753
 Doctorate      >50K     0.740920
                <=50K    0.259080
 HS-grad        <=50K    0.840491
                >50K     0.159509
 Masters        >50K     0.556587
                <=50K    0.443413
 Preschool      <=50K    1.000000
                >50K     0.000000
 Prof-school    >50K     0.734375
                <=50K    0.265625
 Some-college   <=50K    0.809765
                >50K     0.190235
Name: salary, dtype: float64
```

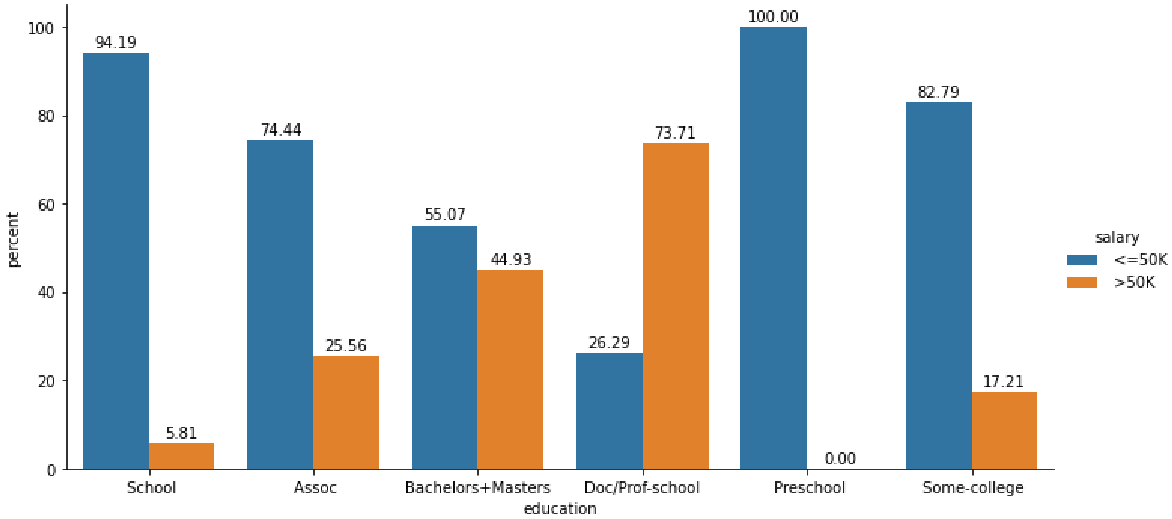
In [91]:

```
mappings={' 10th':' School',' 11th':' School',' 12th':' School',' 1st-4th':' School',' 5th-
df.replace(mappings,inplace=True)
```

In [92]:

```
x1='education'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.savefig('educplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [93]:

```
df.groupby('race')['salary'].value_counts(normalize=True)
```

Out[93]:

```
race                  salary
 Amer-Indian-Eskimo    <=50K     0.884244
                       >50K      0.115756
 Asian-Pac-Islander    <=50K     0.734360
                       >50K      0.265640
 Black                 <=50K     0.876120
                       >50K      0.123880
 Other                 <=50K     0.907749
                       >50K      0.092251
 White                 <=50K     0.744140
                       >50K      0.255860
Name: salary, dtype: float64
```

As we can see the fraction of greater salary holders is similar among white and asians and all other categories have also similar lower fraction of >50K salary.

In [94]:

```
mappings={' Amer-Indian-Eskimo':' Other',' Asian-Pac-Islander':'White+Asian',' Black':' Oth
df['race'].replace(mappings,inplace=True)
df.groupby('race')['salary'].value_counts(normalize=True)
```

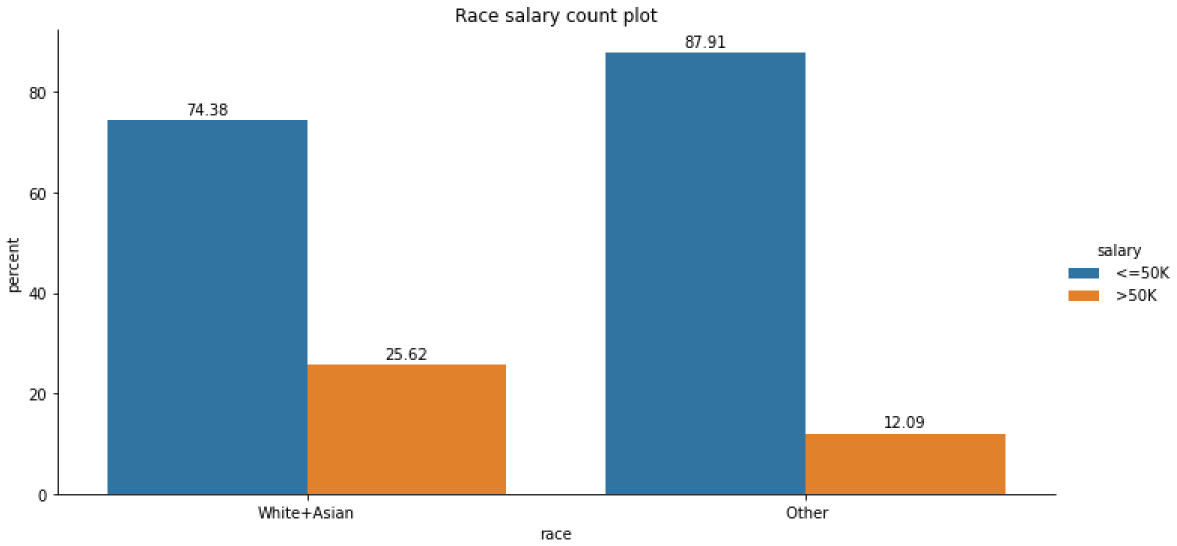
Out[94]:

```
race          salary
White+Asian    <=50K     0.743788
               >50K      0.256212
 Other         <=50K     0.879115
               >50K      0.120885
Name: salary, dtype: float64
```

In [95]:

```
x1='race'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Race salary count plot')
plt.savefig('racecplot2.png')
```

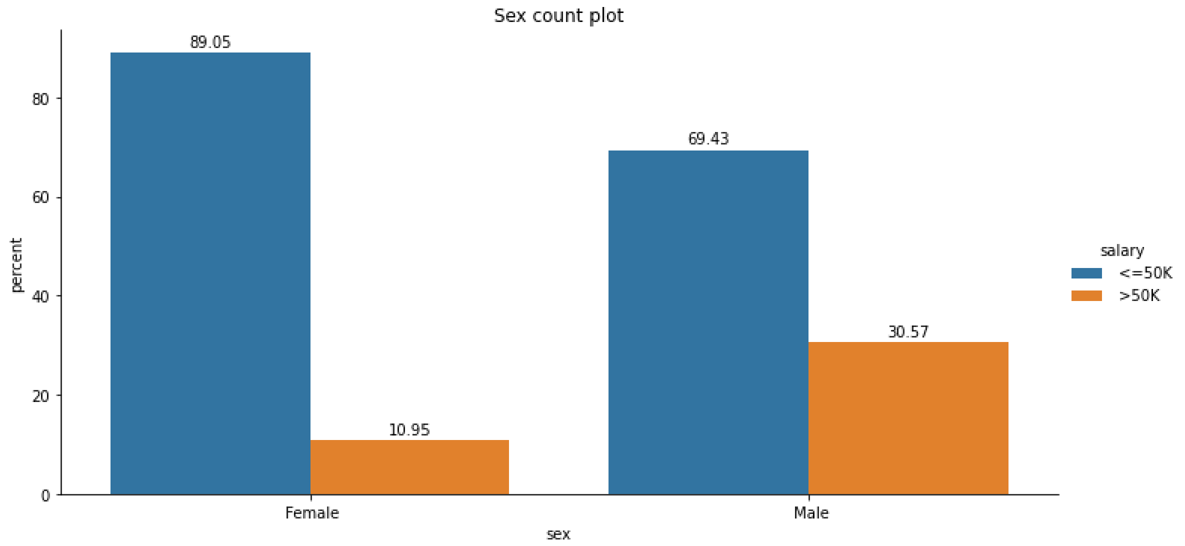
```
<Figure size 1080x504 with 0 Axes>
```



In [96]:

```
x1='sex'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Sex count plot')
plt.savefig('sexcplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [97]:

```
plt.figure(figsize=[10,5])
sns.histplot(data=df,x='age',hue='salary',multiple='dodge',element='poly',fill=False,stat='
plt.title('Age salary count plot')
plt.savefig('Agecplot1.png')
```

Similarly we can discretize the age and hours per week variable as intervals according to the above probability dirstibution

In [98]:

```
df['age']=pd.cut(df.age,bins=[i for i in range(0,160,40)],labels=[str(i)+'-'+str(i+40) for
```

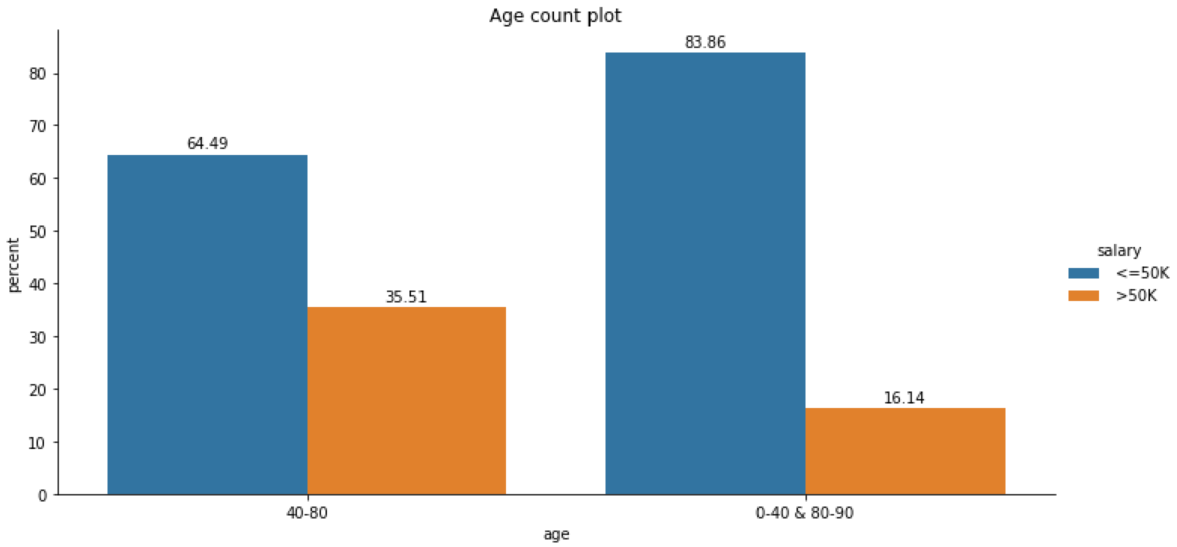
In [99]:

```
mapping={'80-120':'0-40 & 80-90','0-40':'0-40 & 80-90'}
df['age'].replace(mapping,inplace=True)
```

In [100]:

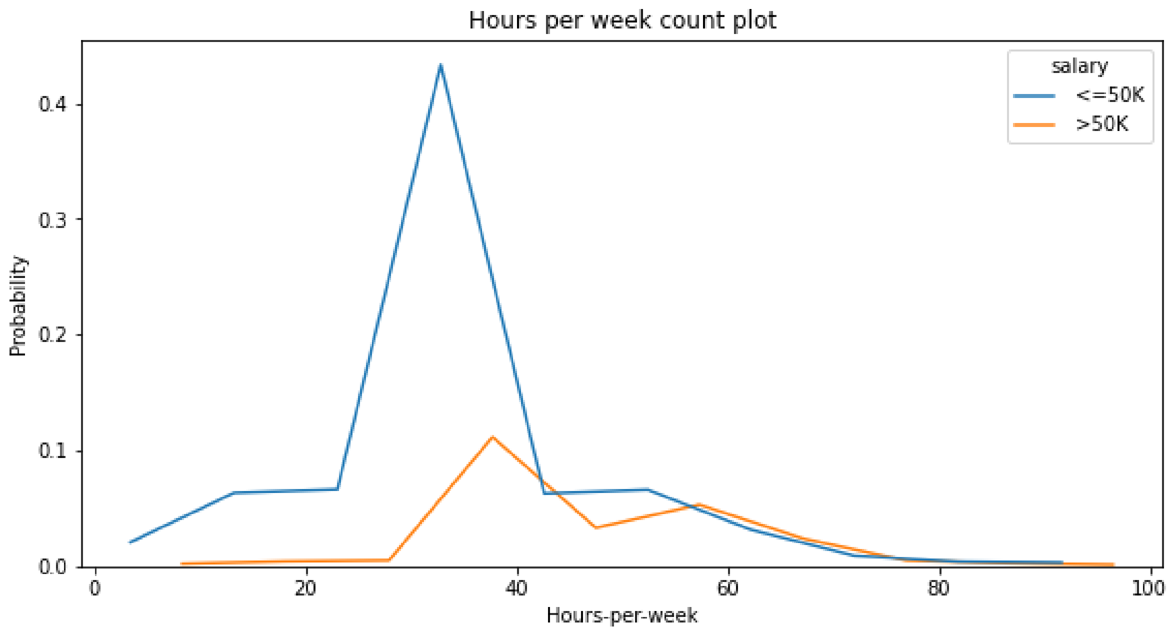
```
x1='age'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Age count plot')
plt.savefig('agecplot2.png')
```

<Figure size 1080x504 with 0 Axes>



In [101]:

```
plt.figure(figsize=[10,5])
sns.histplot(data=df,x='Hours-per-week',hue='salary',bins=10,multiple='dodge',element='poly
plt.title('Hours per week count plot')
plt.savefig('hourscplot.png')
```



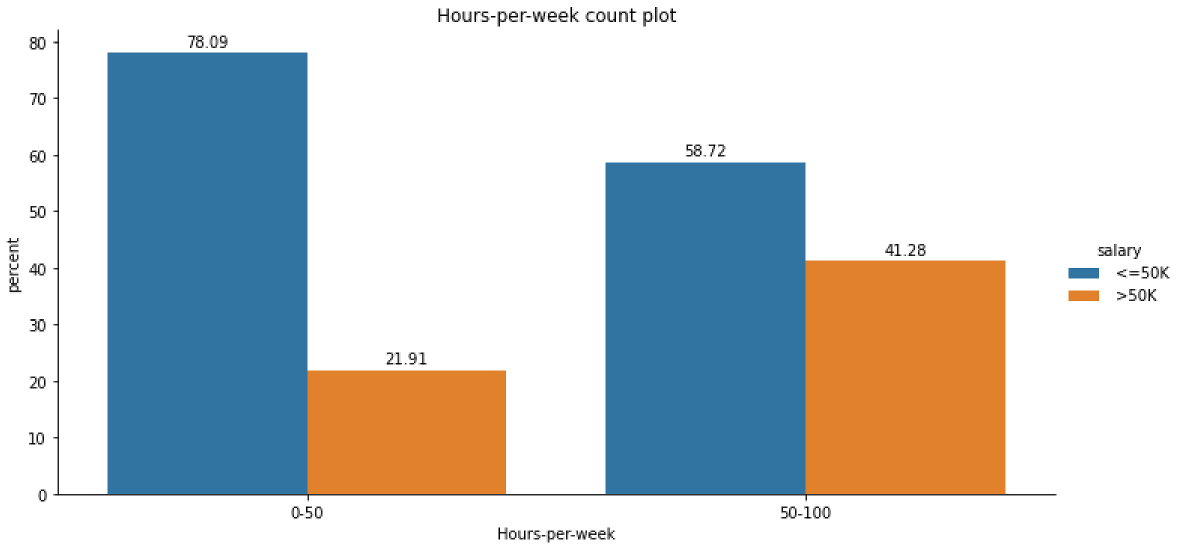
In [102]:

```
df['Hours-per-week']=pd.cut(df['Hours-per-week'],bins=[0,50,100],labels=['0-50','50-100'])
```

In [103]:

```
x1='Hours-per-week'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Hours-per-week count plot')
plt.savefig('Hourscplot2.png')
```

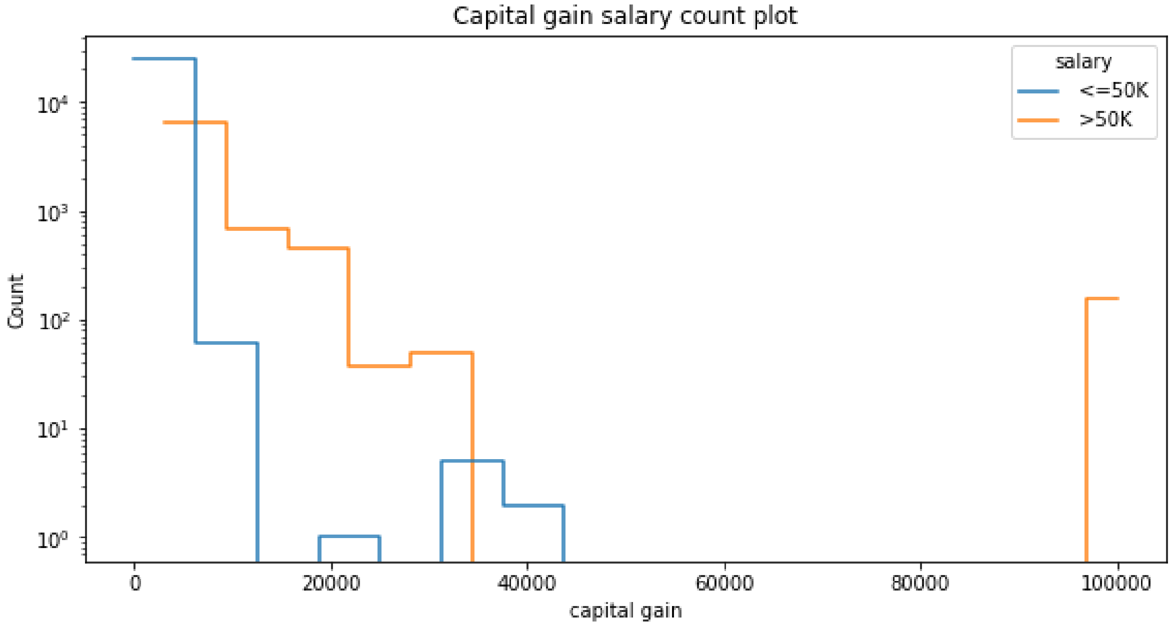
```
<Figure size 1080x504 with 0 Axes>
```



Now let us look at the capital gain and capital loss dirstibutions and proceed to discretize the same

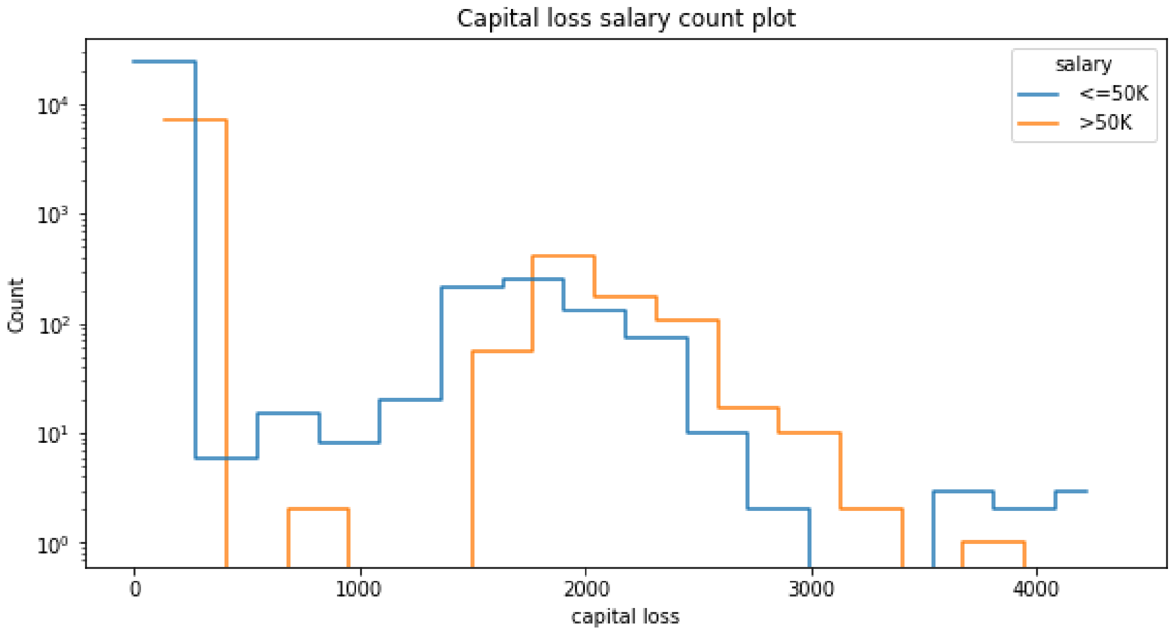
In [104]:

```
plt.figure(figsize=[10,5])
sns.histplot(data=df,x='capital gain',hue='salary',multiple='dodge',element='step',fill=Fal
plt.title('Capital gain salary count plot')
plt.savefig('Capitalgain.png')
```



In [105]:

```
plt.figure(figsize=[10,5])
sns.histplot(data=df,x='capital loss',hue='salary',multiple='dodge',element='step',fill=Fal
plt.title('Capital loss salary count plot')
plt.savefig('Capitalloss.png')
```



In [106]:

```
cap_loss_bins=[0,1000,4000,5000]
cap_gain_bins=[0,5000,100000]
cap_loss_labels=[str(cap_loss_bins[i])+'-'+str(cap_loss_bins[i+1]) for i in range(len(cap_l
cap_gain_labels=[str(cap_gain_bins[i])+'-'+str(cap_gain_bins[i+1]) for i in range(len(cap_g
```

In [107]:

```
df['capital gain']=pd.cut(df['capital gain'],bins=cap_gain_bins,labels=cap_gain_labels,incl
```

In [108]:

```
df['capital loss']=pd.cut(df['capital loss'],bins=cap_loss_bins,labels=cap_loss_labels,incl
```

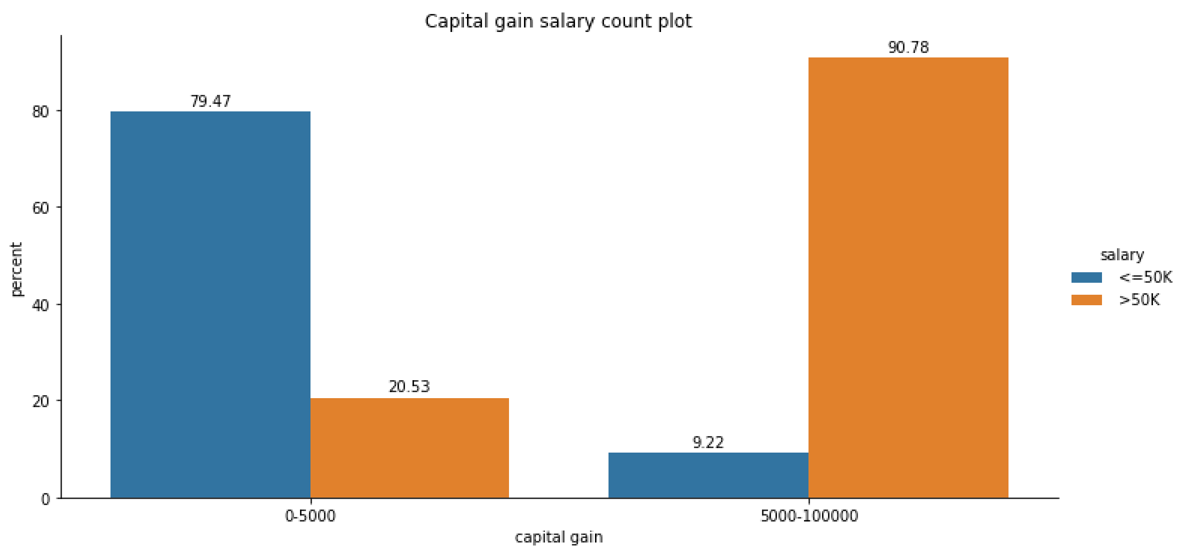
In [109]:

```
mappings={'4000-5000':'0-1000 & 4000-5000','0-1000':'0-1000 & 4000-5000'}
df['capital loss'].replace(mappings,inplace=True)
```

In [110]:

```
x1='capital gain'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Capital gain salary count plot')
plt.savefig('Capitalgain2.png')
```

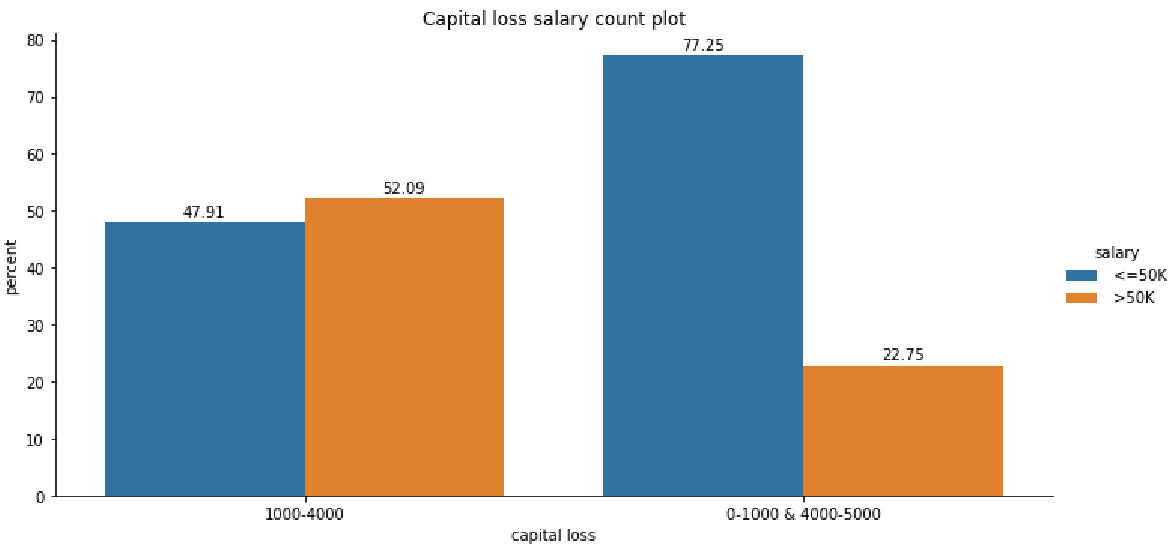
<Figure size 1080x504 with 0 Axes>



In [111]:

```
x1='capital loss'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Capital loss salary count plot')
plt.savefig('Capitalloss2.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [112]:

```python
df.groupby('marital status')['salary'].value_counts(normalize=True)
```

Out[112]:

```
marital status          salary
 Divorced                <=50K     0.895791
                         >50K      0.104209
 Married-AF-spouse       <=50K     0.565217
                         >50K      0.434783
 Married-civ-spouse      <=50K     0.553152
                         >50K      0.446848
 Married-spouse-absent   <=50K     0.918660
                         >50K      0.081340
 Never-married           <=50K     0.954039
                         >50K      0.045961
 Separated               <=50K     0.935610
                         >50K      0.064390
 Widowed                 <=50K     0.914401
                         >50K      0.085599
Name: salary, dtype: float64
```

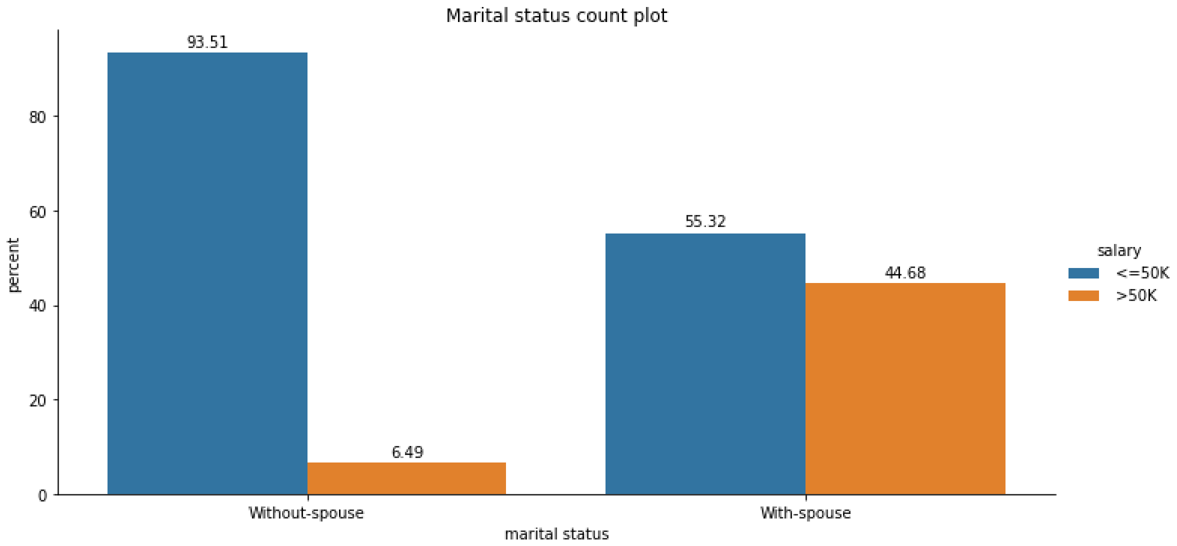
In [113]:

```python
mapping={' Divorced':'Without-spouse',' Married-AF-spouse':'With-spouse',' Married-civ-spou
df['marital status'].replace(mapping,inplace=True)
```

In [114]:

```
x1='marital status'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Marital status count plot')
plt.savefig('maritalcplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [115]:

```
df.groupby('Occupation')['salary'].value_counts(normalize=True)
```

Out[115]:

```
Occupation          salary
 Adm-clerical        <=50K     0.865517
                     >50K      0.134483
 Armed-Forces        <=50K     0.888889
                     >50K      0.111111
 Craft-repair        <=50K     0.773359
                     >50K      0.226641
 Exec-managerial     <=50K     0.515986
                     >50K      0.484014
 Farming-fishing     <=50K     0.884306
                     >50K      0.115694
 Handlers-cleaners   <=50K     0.937226
                     >50K      0.062774
 Machine-op-inspct   <=50K     0.875125
                     >50K      0.124875
 Other-service       <=50K     0.958422
                     >50K      0.041578
 Priv-house-serv     <=50K     0.993289
                     >50K      0.006711
 Prof-specialty      <=50K     0.550966
                     >50K      0.449034
 Protective-serv     <=50K     0.674884
                     >50K      0.325116
 Sales               <=50K     0.730685
                     >50K      0.269315
 Tech-support        <=50K     0.695043
                     >50K      0.304957
 Transport-moving    <=50K     0.799624
                     >50K      0.200376
Unknown              <=50K     0.896365
                     >50K      0.103635
Name: salary, dtype: float64
```

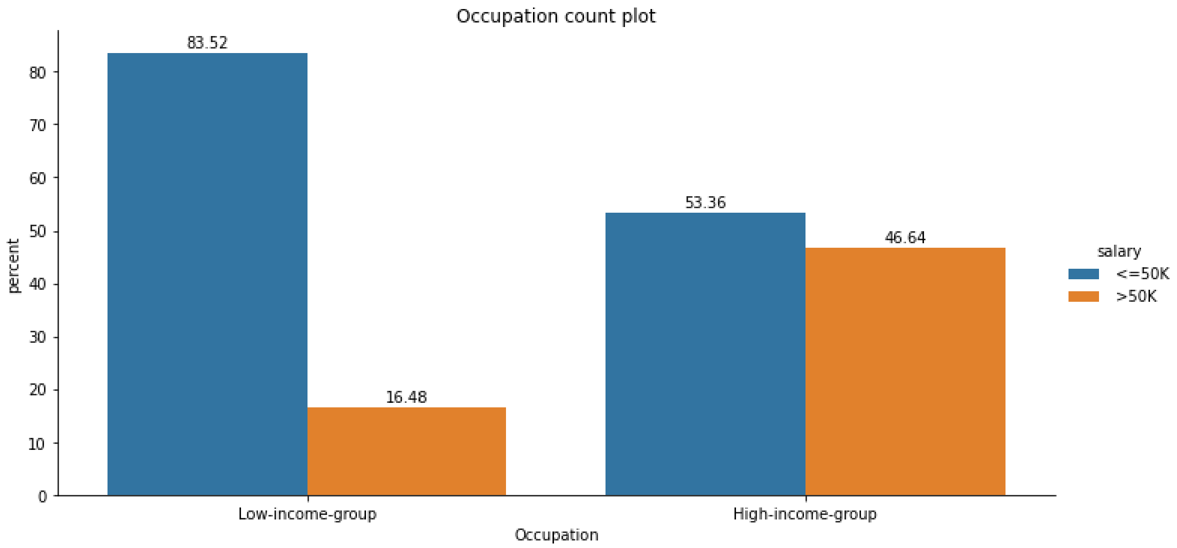
In [116]:

```
mappings={' Adm-clerical':'Low-income-group',' Armed-Forces':'Low-income-group',' Farming-f
df['Occupation'].replace(mappings,inplace=True)
```

In [117]:

```
x1='Occupation'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Occupation count plot')
plt.savefig('occcplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [118]:

```
df.groupby('relationship')['salary'].value_counts(normalize=True)
```

Out[118]:

```
relationship     salary
 Husband          <=50K     0.551429
                  >50K      0.448571
 Not-in-family    <=50K     0.896930
                  >50K      0.103070
 Other-relative   <=50K     0.962283
                  >50K      0.037717
 Own-child        <=50K     0.986780
                  >50K      0.013220
 Unmarried        <=50K     0.936738
                  >50K      0.063262
 Wife             <=50K     0.524872
                  >50K      0.475128
Name: salary, dtype: float64
```

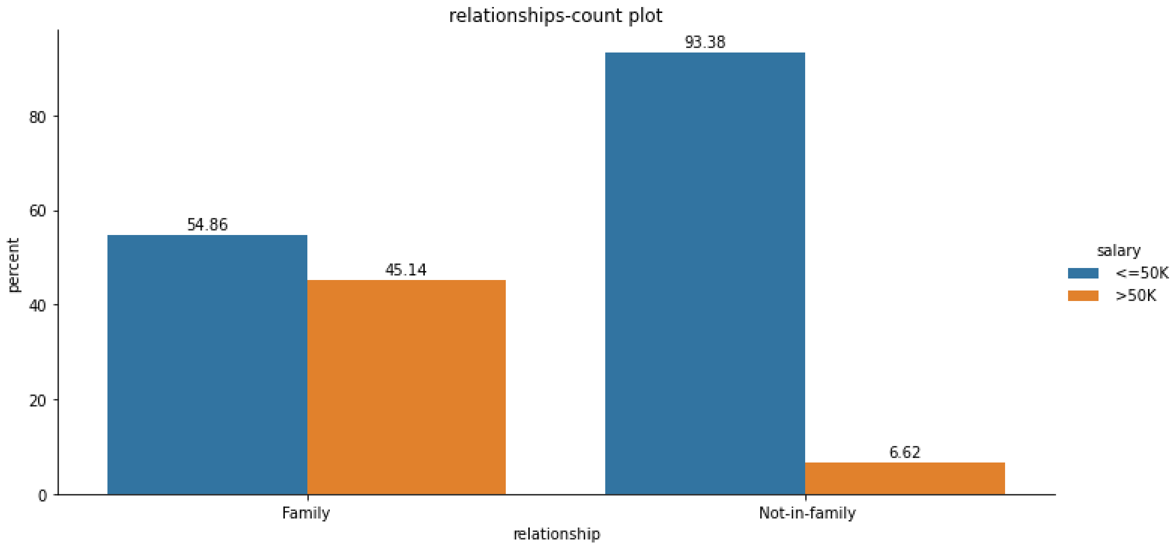
In [119]:

```
mappings={' Husband':'Family',' Wife':'Family',' Other-relative':' Not-in-family',' Unmarri
df['relationship'].replace(mappings,inplace=True)
```

In [120]:

```
x1='relationship'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('relationships-count plot')
plt.savefig('relcplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



In [121]:

```
df['Native-country'].value_counts(sort=True,ascending=False,normalize=True)
```

Out[121]:

```
 United-States                 0.895857
 Mexico                        0.019748
Unknown                        0.017905
 Philippines                   0.006081
 Germany                       0.004207
 Canada                        0.003716
 Puerto-Rico                   0.003501
 El-Salvador                   0.003255
 India                         0.003071
 Cuba                          0.002918
 England                       0.002764
 Jamaica                       0.002488
 South                         0.002457
 China                         0.002303
 Italy                         0.002242
 Dominican-Republic            0.002150
 Vietnam                       0.002058
 Guatemala                     0.001966
 Japan                         0.001904
 Poland                        0.001843
 Columbia                      0.001812
 Taiwan                        0.001566
 Haiti                         0.001351
 Iran                          0.001321
 Portugal                      0.001136
 Nicaragua                     0.001044
 Peru                          0.000952
 France                        0.000891
 Greece                        0.000891
 Ecuador                       0.000860
 Ireland                       0.000737
 Hong                          0.000614
 Trinadad&Tobago               0.000584
 Cambodia                      0.000584
 Thailand                      0.000553
 Laos                          0.000553
 Yugoslavia                    0.000491
 Outlying-US(Guam-USVI-etc)    0.000430
 Hungary                       0.000399
 Honduras                      0.000399
 Scotland                      0.000369
 Holand-Netherlands            0.000031
Name: Native-country, dtype: float64
```

Now let us split categories based on whether or not it is US

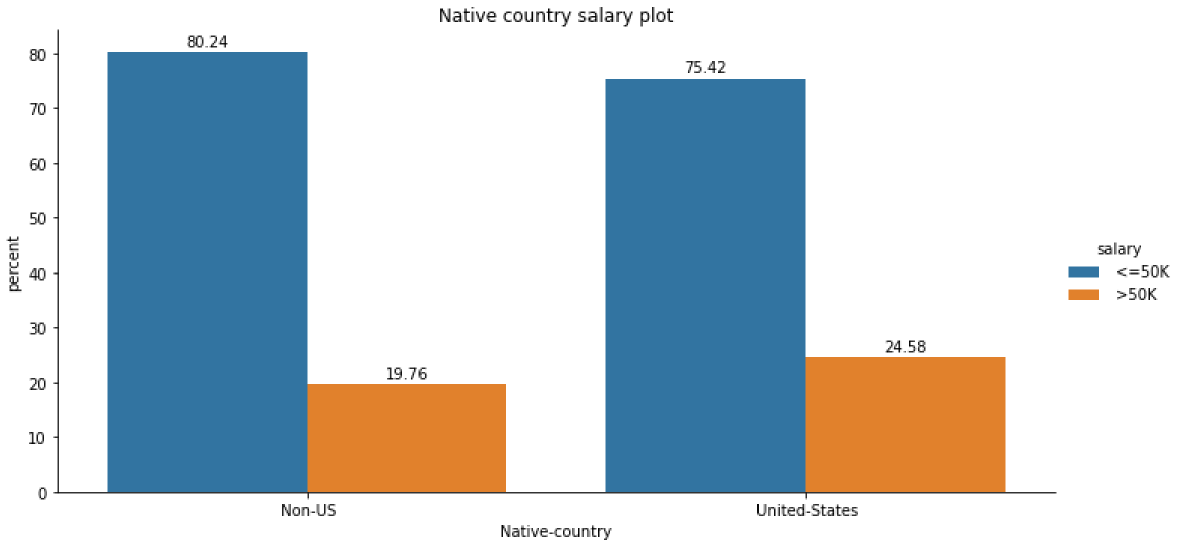
In [122]:

```
def func(x):
    if x!=' United-States':
        return ' Non-US'
    else:
        return x
df['Native-country']=df['Native-country'].apply(func)
```

In [123]:

```
x1='Native-country'
y='salary'
dfp=(df
.groupby(x1)[y]
.value_counts(normalize=True)
.mul(100)
.rename('percent')
.reset_index())
plt.figure(figsize=[15,7])
g=sns.catplot(x=x1,y='percent',hue=y,data=dfp,kind='bar',height=5,aspect=2)
for container in g.ax.containers:
    g.ax.bar_label(container, fmt='%.2f', padding=2)
plt.title('Native country salary plot')
plt.savefig('Nativecplot.png')
```

```
<Figure size 1080x504 with 0 Axes>
```



Feature Extraction

In [124]:

```
X=df.drop(columns=['fnlwgt','salary','education','work class'])# We are choosing categories
                                                               # could clean upto binary fe
                                                             #dropping other input columns
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype   
---  ------          --------------  -----   
 0   age             32561 non-null  category
 1   education-num   32561 non-null  category
 2   marital status  32561 non-null  category
 3   Occupation      32561 non-null  category
 4   relationship    32561 non-null  category
 5   race            32561 non-null  category
 6   sex             32561 non-null  category
 7   capital gain    32561 non-null  category
 8   capital loss    32561 non-null  category
 9   Hours-per-week  32561 non-null  category
 10  Native-country  32561 non-null  object  
dtypes: category(10), object(1)
memory usage: 573.7+ KB
```

In [125]:

```
y=df['salary']
```

Label Encoding for all input categories

In [126]:

```
from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
```

In [127]:

```
X1=pd.DataFrame()
for i in list(X.columns):
    X1[i]=enc.fit_transform(X[i])
X1.head()
```

Out[127]:

	age	education-num	marital status	Occupation	relationship	race	sex	capital gain	capital loss	Hours-per-week	Nativ count
0	0	1	1	1	0	1	1	0	0	0	
1	1	1	0	0	1	1	1	0	0	0	
2	0	0	1	1	0	1	1	0	0	0	
3	1	0	0	1	1	0	1	0	0	0	
4	0	1	0	0	1	0	0	0	0	0	

Train test Split

In [128]:

```
from sklearn.model_selection import train_test_split
```

In [129]:

```
X_train, X_test, y_train, y_test = train_test_split(X1, y, test_size=0.1, random_state=42)
```

Train

In [130]:

```
from sklearn.naive_bayes import BernoulliNB
```

In [131]:

```
Bnb=BernoulliNB()
```

In [132]:

```
Bnb.fit(X_train,y_train)
ypred=Bnb.predict(X_test)
```

Performance

In [133]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [134]:

```
accuracy_score(y_test,ypred)
```

Out[134]:

```
0.7985876573533927
```

In [135]:

```
confusion_matrix(y_test,ypred)
```

Out[135]:

```
array([[1995,  461],
       [ 195,  606]], dtype=int64)
```