1) Prepare a classification model using Naive Bayes for salary data

Data Description:

age -- age of a person
workclass -- A work class is a grouping of work
education -- Education of an individuals
maritalstatus -- Marital status of an individulas

occupation -- occupation of an individuals

relationship --
race -- Race of an Individual
sex -- Gender of an Individual
capitalgain -- profit received from the sale of an investment
capitalloss -- A decrease in the value of a capital asset
hoursperweek -- number of hours work per week
native -- Native of an individual
Salary -- salary of an individual

In [95]:

```
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
from sklearn.preprocessing import normalize, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix





from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, accuracy_score,precision_score,recall_score,f1_score,matt
from sklearn.metrics import confusion_matrix


%matplotlib inline
sns.set_style('darkgrid')

import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
test = pd.read_csv("test.csv")
train = pd.read_csv("train.csv")

train.head()
```

Out[2]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | capitalgain | capitalloss | hoursperv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39 | State-gov | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | |
| 1 | 50 | Self-emp-not-inc | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | |
| 2 | 38 | Private | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | |
| 3 | 53 | Private | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | |
| 4 | 28 | Private | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | |

In [3]:

```
salary_data = pd.concat([train,test], axis = 0).reset_index(drop = True)
salary_data.head()
```

Out[3]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | capitalgain | capitalloss | hoursperv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 39 | State-gov | Bachelors | 13 | Never-married | Adm-clerical | Not-in-family | White | Male | 2174 | 0 | |
| 1 | 50 | Self-emp-not-inc | Bachelors | 13 | Married-civ-spouse | Exec-managerial | Husband | White | Male | 0 | 0 | |
| 2 | 38 | Private | HS-grad | 9 | Divorced | Handlers-cleaners | Not-in-family | White | Male | 0 | 0 | |
| 3 | 53 | Private | 11th | 7 | Married-civ-spouse | Handlers-cleaners | Husband | Black | Male | 0 | 0 | |
| 4 | 28 | Private | Bachelors | 13 | Married-civ-spouse | Prof-specialty | Wife | Black | Female | 0 | 0 | |

In [4]:

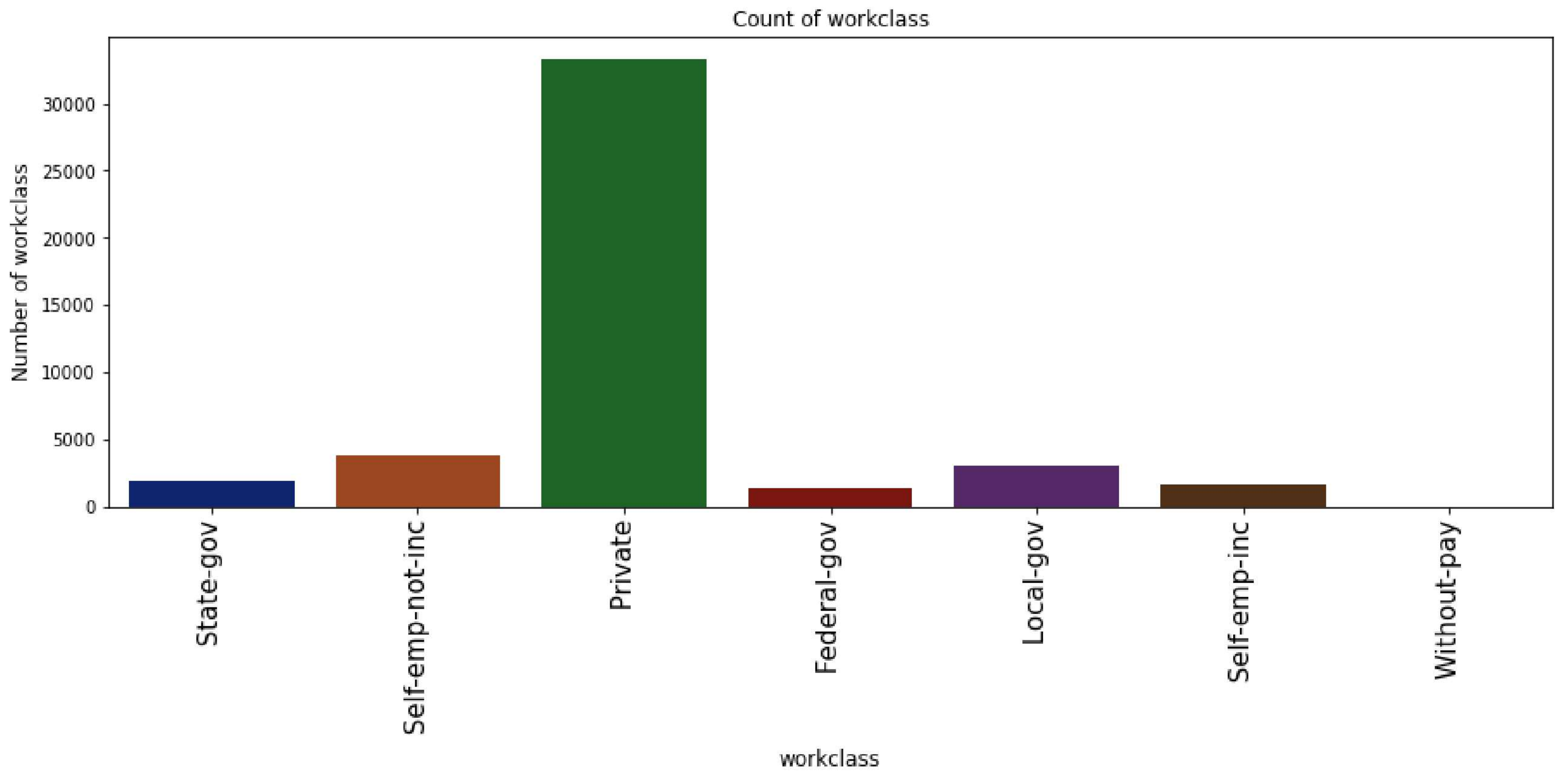
```
numerical_features =  salary_data.describe(include=["int64"]).columns

print(list(numerical_features))

categorical_features =  salary_data.describe(include=["object"]).columns

print(list(categorical_features))
```

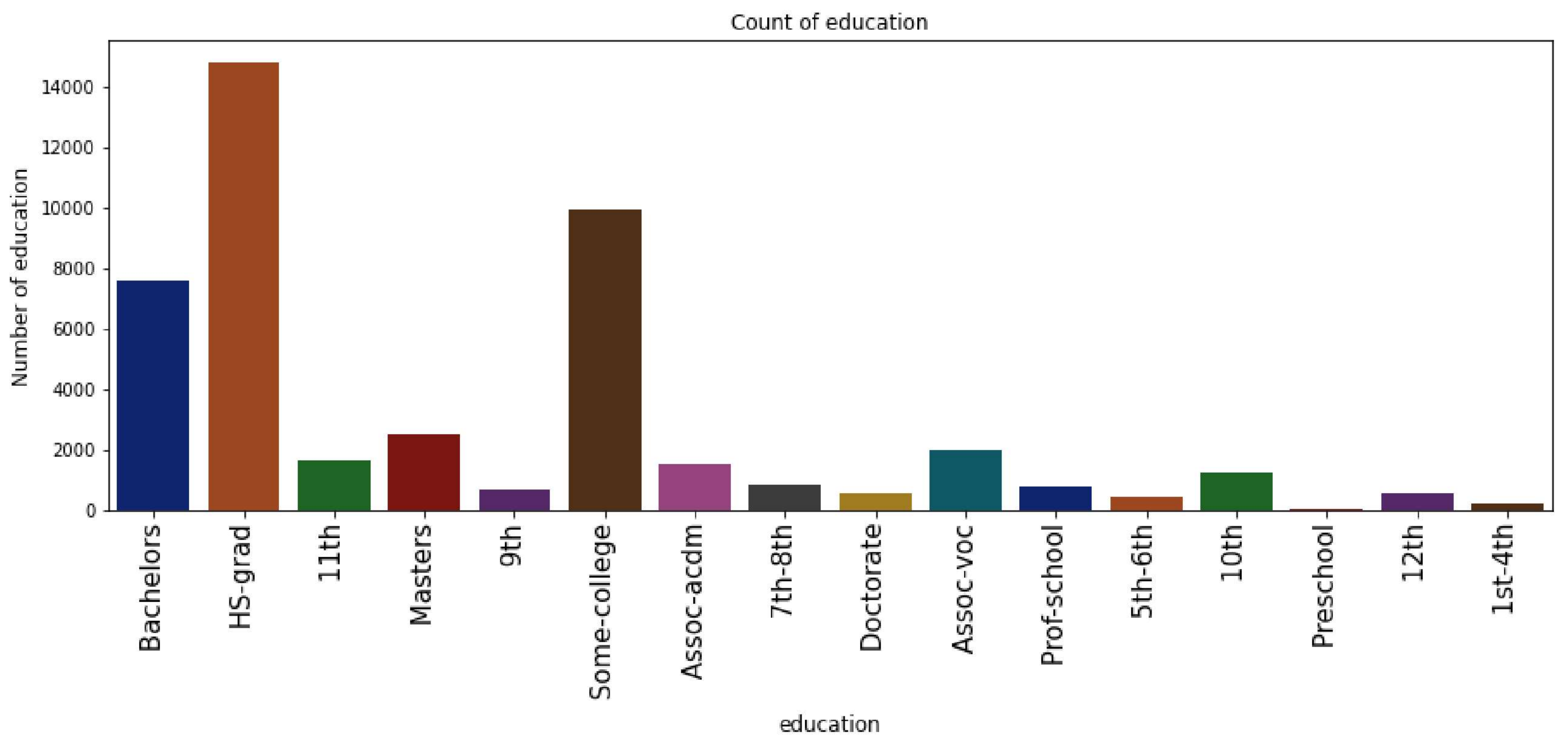
```
['age', 'educationno', 'capitalgain', 'capitalloss', 'hoursperweek']
['workclass', 'education', 'maritalstatus', 'occupation', 'relationship', 'race', 'sex', 'native', 'Salar
y']
```

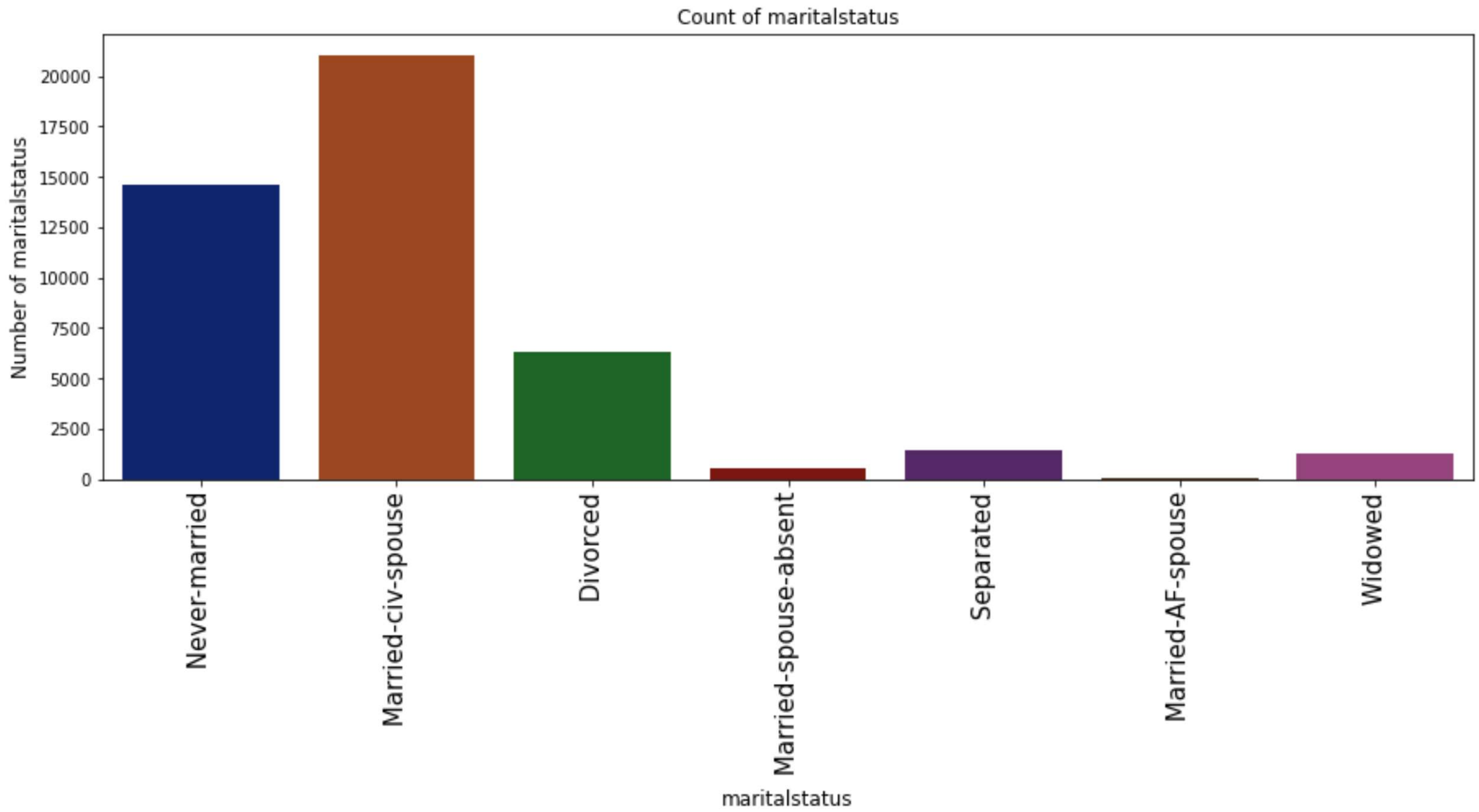
In [5]:

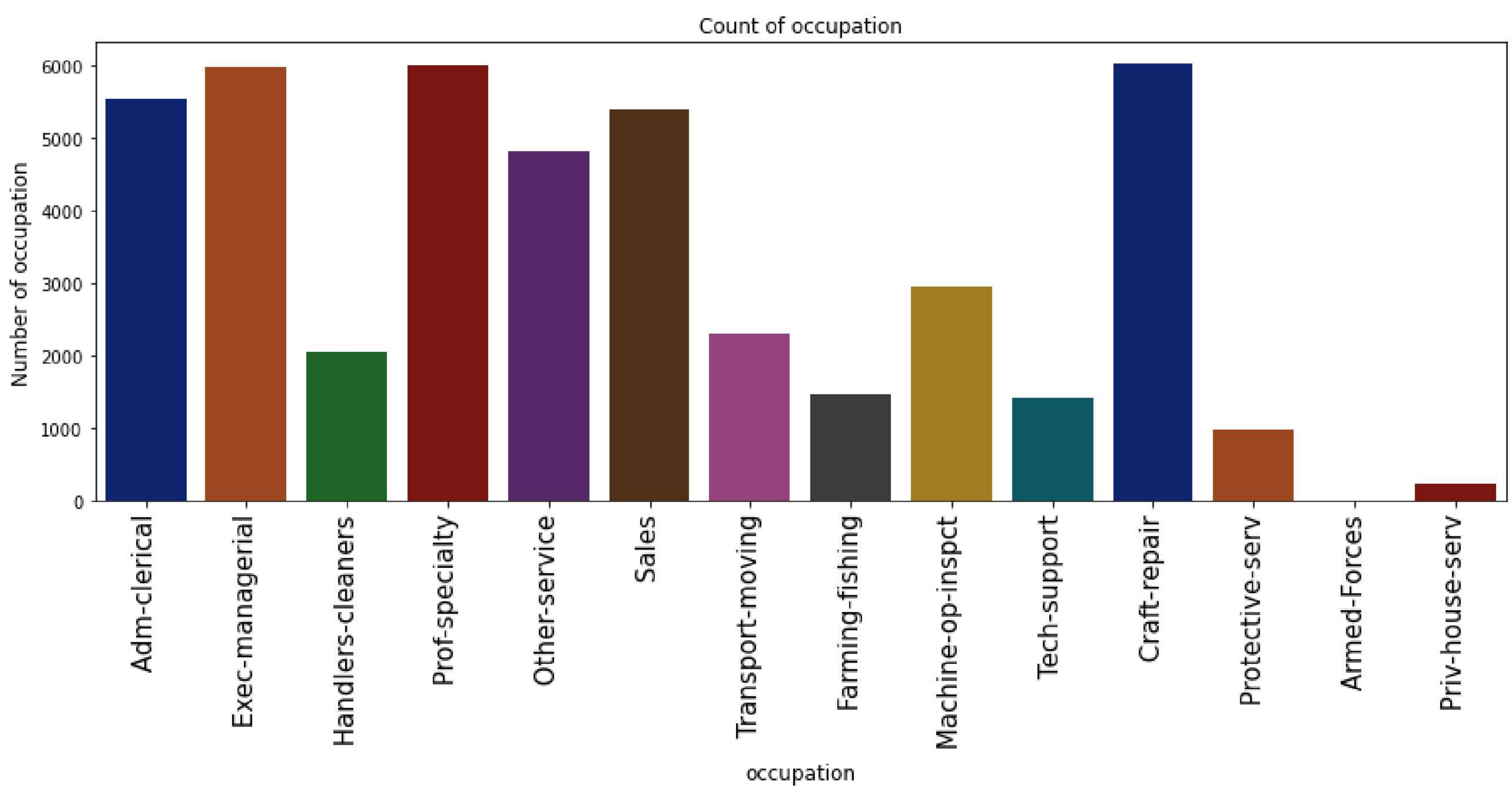
```
print(categorical_features)



for idx, column in enumerate(categorical_features):
    plt.figure(figsize=(15, 5))
    df = salary_data.copy()
    unique = df[column].value_counts(ascending=True);

    #plt.subplot(1, len(categorical_features), idx+1)
    plt.title("Count of "+ column)
    sns.countplot(data=salary_data, x=column,palette = "dark")
    #plt.bar(unique.index, unique.values);
    plt.xticks(rotation = 90, size = 15)

    plt.xlabel(column, fontsize=12)
    plt.ylabel("Number of "+ column, fontsize=12)
    plt.show()

#plt.tight_layout()
```

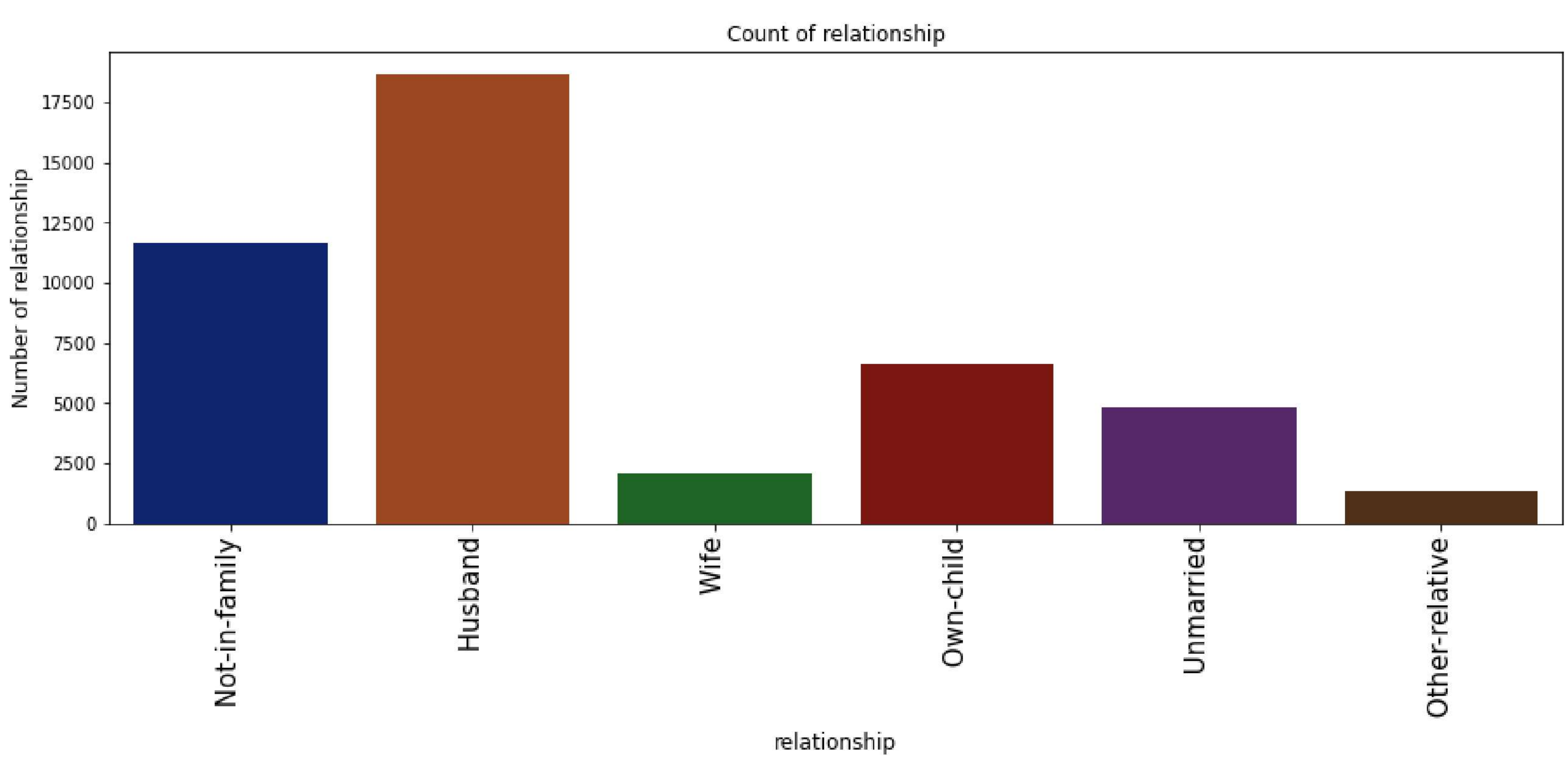
```
Index(['workclass', 'education', 'maritalstatus', 'occupation', 'relationship',
       'race', 'sex', 'native', 'Salary'],
      dtype='object')
```

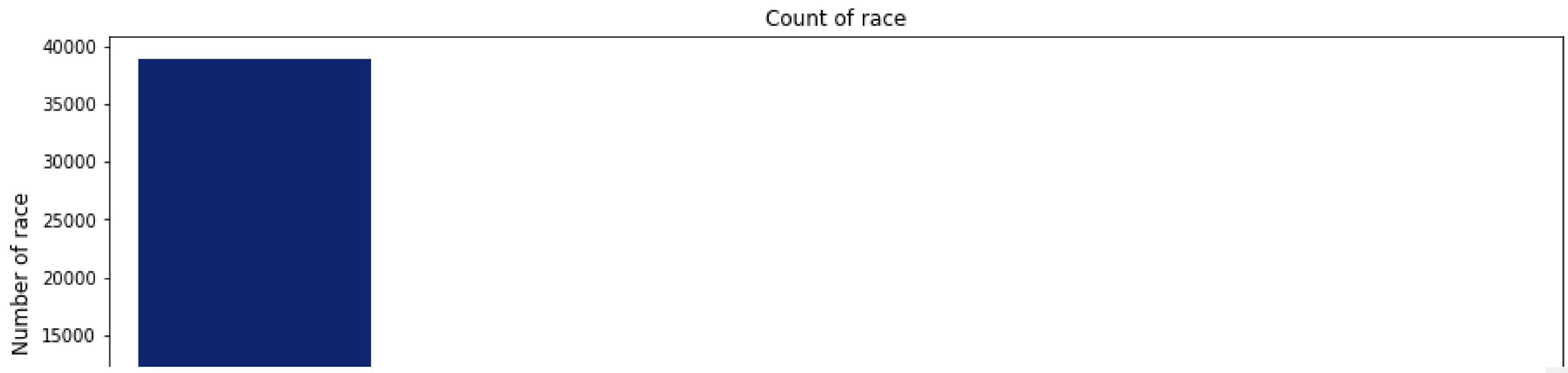


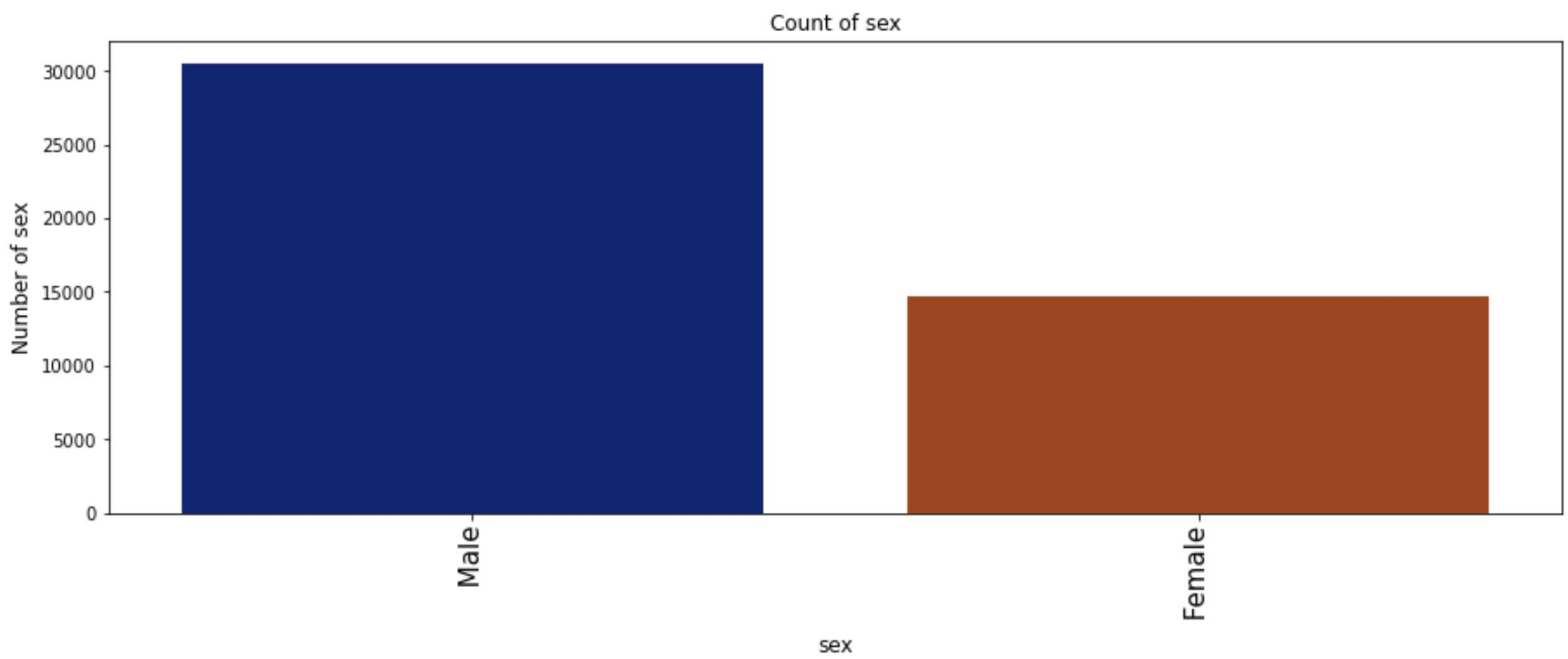



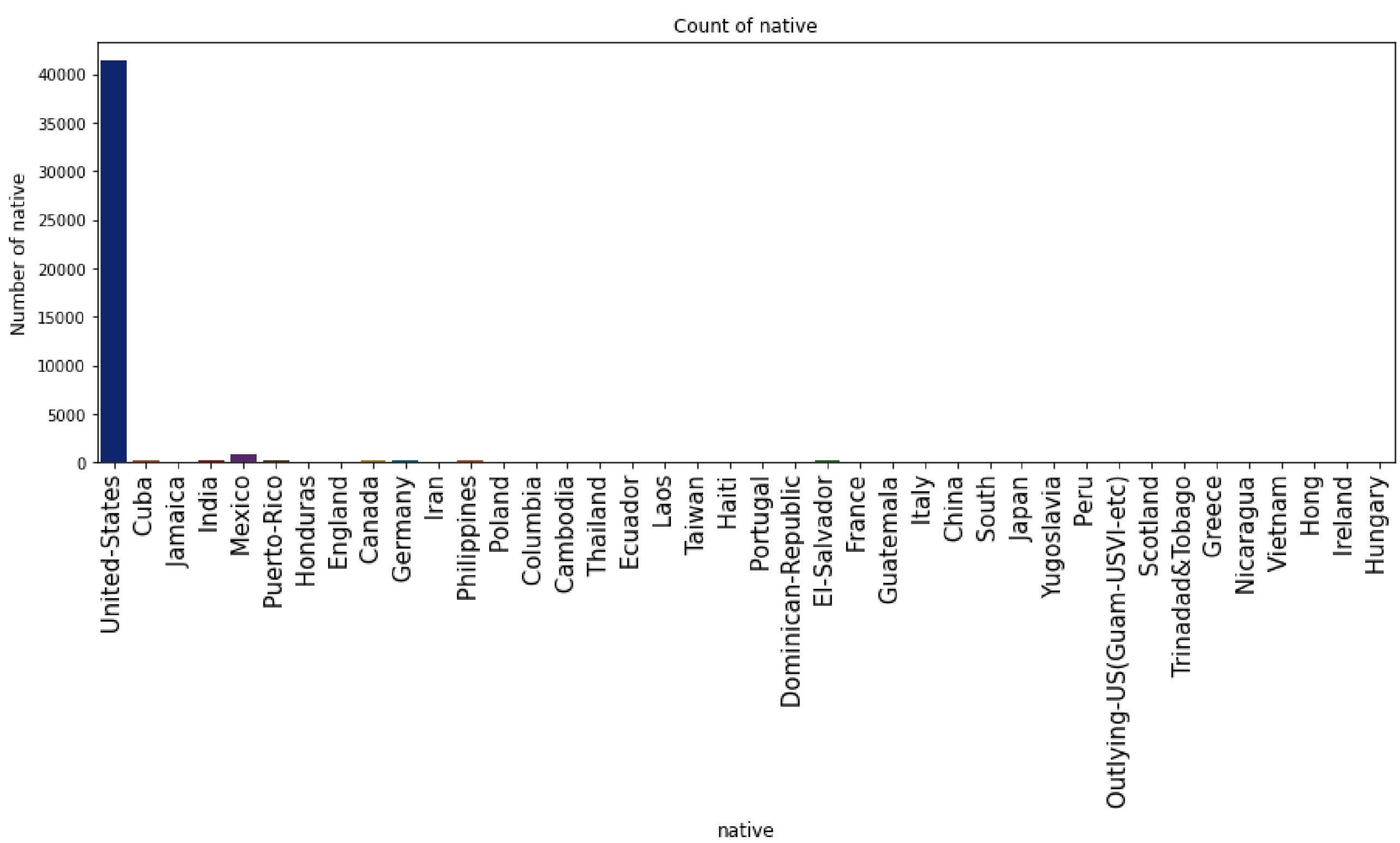

Count of maritalstatus
Number of maritalstatus
0
2500
5000
7500
10000
12500
15000
17500
20000
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Married-AF-spouse
Widowed
maritalstatus

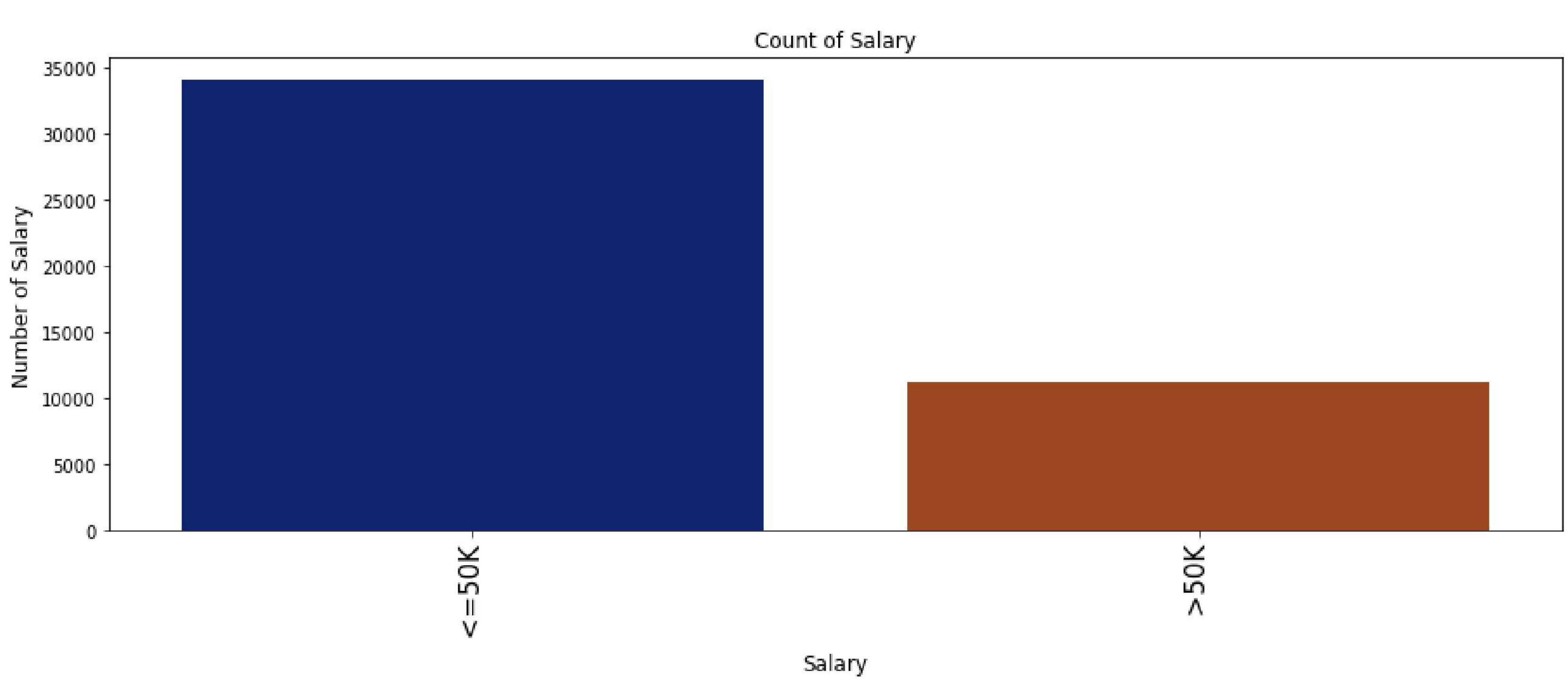

Count of occupation
Number of occupation
0
1000
2000
3000
4000
5000
6000
Adm-clerical
Exec-managerial
Handlers-cleaners
Prof-specialty
Other-service
Sales
Transport-moving
Farming-fishing
Machine-op-inspct
Tech-support
Craft-repair
Protective-serv
Armed-Forces
Priv-house-serv
occupation


Count of relationship
Number of relationship
0
2500
5000
7500
10000
12500
15000
17500
Not-in-family
Husband
Wife
Own-child
Unmarried
Other-relative
relationship


Count of race
40000
35000
30000
25000
20000
15000
Number of race

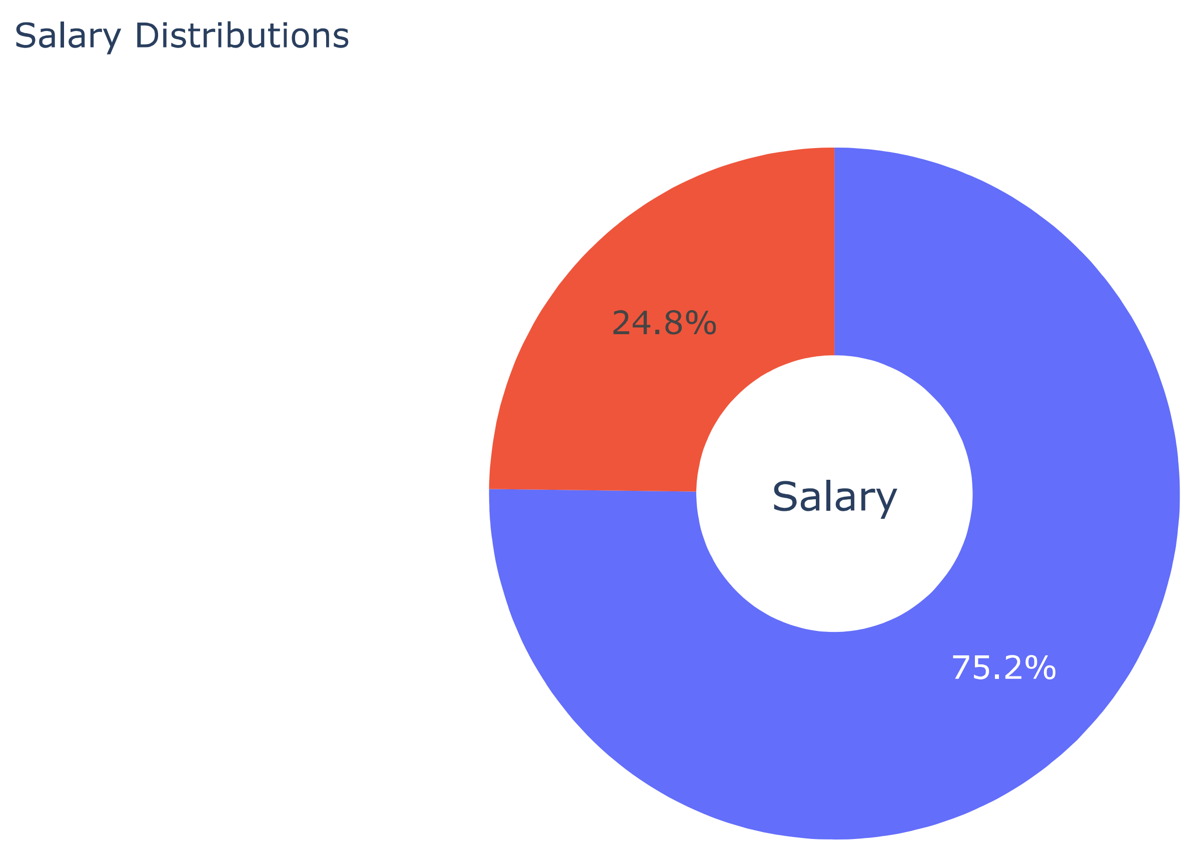

Count of sex
30000
25000
20000
15000
10000
5000
0
Number of sex
Male
Female
sex


Count of native
40000
35000
30000
25000
20000
15000
10000
5000
0
Number of native
United-States
Cuba
Jamaica
India
Mexico
Puerto-Rico
Honduras
England
Canada
Germany
Iran
Philippines
Poland
Columbia
Cambodia
Thailand
Ecuador
Laos
Taiwan
Haiti
Portugal
Dominican-Republic
El-Salvador
France
Guatemala
Italy
China
South
Japan
Yugoslavia
Peru
Outlying-US(Guam-USVI-etc)
Scotland
Trinadad&Tobago
Greece
Nicaragua
Vietnam
Hong
Ireland
Hungary
native


Count of Salary
35000
30000
25000
20000
15000
10000
5000
0
Number of Salary
<=50K
>50K
Salary

In [8]:

```
type_ = [' <=50K', ' >50K']
fig = make_subplots(rows=1, cols=1)

fig.add_trace(go.Pie(labels=type_, values=salary_data['Salary'].value_counts(), name="Salary"))

# Use `hole` to create a donut-like pie chart
fig.update_traces(hole=.4, hoverinfo="label+percent+name", textfont_size=16)

fig.update_layout(
    title_text="Salary Distributions",
    # Add annotations in the center of the donut pies.
    annotations=[dict(text='Salary', x=0.5, y=0.5, font_size=20, showarrow=False)])
fig.show()
```




In [27]:

```
salary_data.Salary[salary_data.Salary == ' <=50K'].groupby(by = salary_data.sex).count()
```

Out[27]:

```
sex
 Female    13025
 Male      20988
Name: Salary, dtype: int64
```

In [28]:

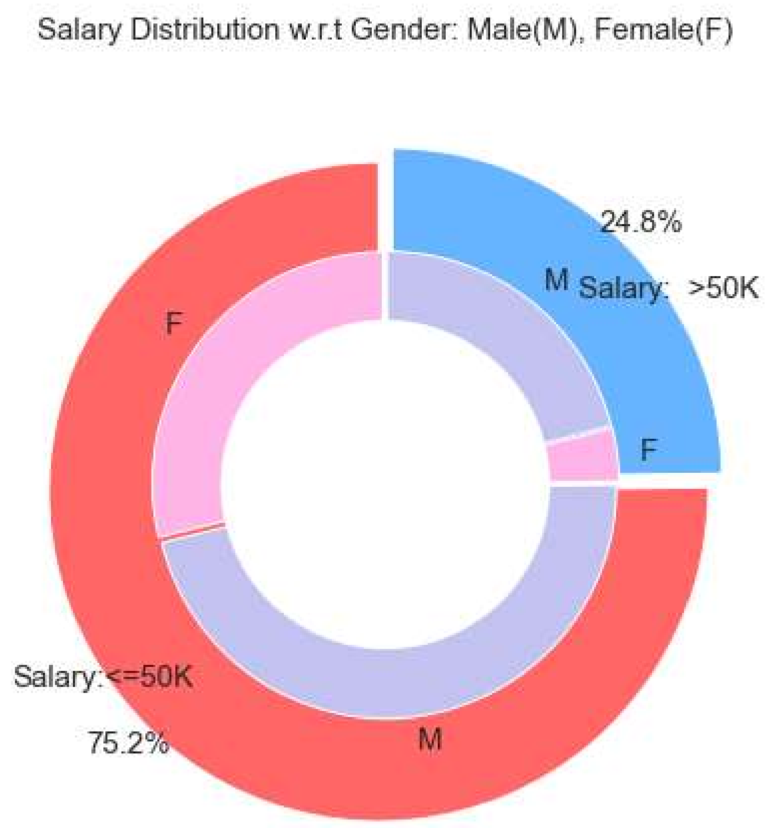
```
salary_data.Salary[salary_data.Salary == ' >50K'].groupby(by = salary_data.sex).count()
```

Out[28]:

```
sex
 Female    1669
 Male      9539
Name: Salary, dtype: int64
```

In [29]:

```
plt.figure(figsize=(6, 6))
labels =["Salary:<=50K", "Salary:  >50K"]
values = [salary_data.Salary[salary_data.Salary == ' <=50K'].groupby(by = salary_data.sex).count().sum(),
         salary_data.Salary[salary_data.Salary == ' >50K'].groupby(by = salary_data.sex).count().sum()]
labels_gender = ["F","M","F","M"]
sizes_gender = [13025,20988 , 1669,9539]
colors = ['#ff6666', '#66b3ff']
colors_gender = ['#ffb3e6','#c2c2f0','#ffb3e6', '#c2c2f0']
explode = (0.3,0.3)
explode_gender = (0.1,0.1,0.1,0.1)
textprops = {"fontsize":15}
#Plot
plt.pie(values, labels=labels,autopct='%1.1f%%',pctdistance=1.08, labeldistance=0.8,colors=colors, startangl
plt.pie(sizes_gender,labels=labels_gender,colors=colors_gender,startangle=90, explode=explode_gender,radius=
#Draw circle
centre_circle = plt.Circle((0,0),5,color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.title('Salary Distribution w.r.t Gender: Male(M), Female(F)', fontsize=15, y=1.1)

# show plot

plt.axis('equal')
plt.tight_layout()
plt.show()
```



In [30]:
```
from sklearn.preprocessing import LabelEncoder
train_data = train.apply(LabelEncoder().fit_transform)
train_data.head()
```

Out[30]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | capitalgain | capitalloss | hoursperweek |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 22 | 5 | 9 | 12 | 4 | 0 | 1 | 4 | 1 | 24 | 0 | 39 |
| **1** | 33 | 4 | 9 | 12 | 2 | 3 | 0 | 4 | 1 | 0 | 0 | 12 |
| **2** | 21 | 2 | 11 | 8 | 0 | 5 | 1 | 4 | 1 | 0 | 0 | 39 |
| **3** | 36 | 2 | 1 | 6 | 2 | 5 | 0 | 2 | 1 | 0 | 0 | 39 |
| **4** | 11 | 2 | 9 | 12 | 2 | 9 | 5 | 2 | 0 | 0 | 0 | 39 |

In [31]:
```
test_data = test.apply(LabelEncoder().fit_transform)
test_data.head()
```

Out[31]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | capitalgain | capitalloss | hoursperweek |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 8 | 2 | 1 | 6 | 4 | 6 | 3 | 2 | 1 | 0 | 0 | 39 |
| **1** | 21 | 2 | 11 | 8 | 2 | 4 | 0 | 4 | 1 | 0 | 0 | 49 |
| **2** | 11 | 1 | 7 | 11 | 2 | 10 | 0 | 4 | 1 | 0 | 0 | 39 |
| **3** | 27 | 2 | 15 | 9 | 2 | 6 | 0 | 2 | 1 | 87 | 0 | 39 |
| **4** | 17 | 2 | 0 | 5 | 4 | 7 | 1 | 4 | 1 | 0 | 0 | 29 |

In [32]:
```
data_copy = pd.concat([train_data,test_data], axis = 0).reset_index(drop = True)
data_copy.head()
```

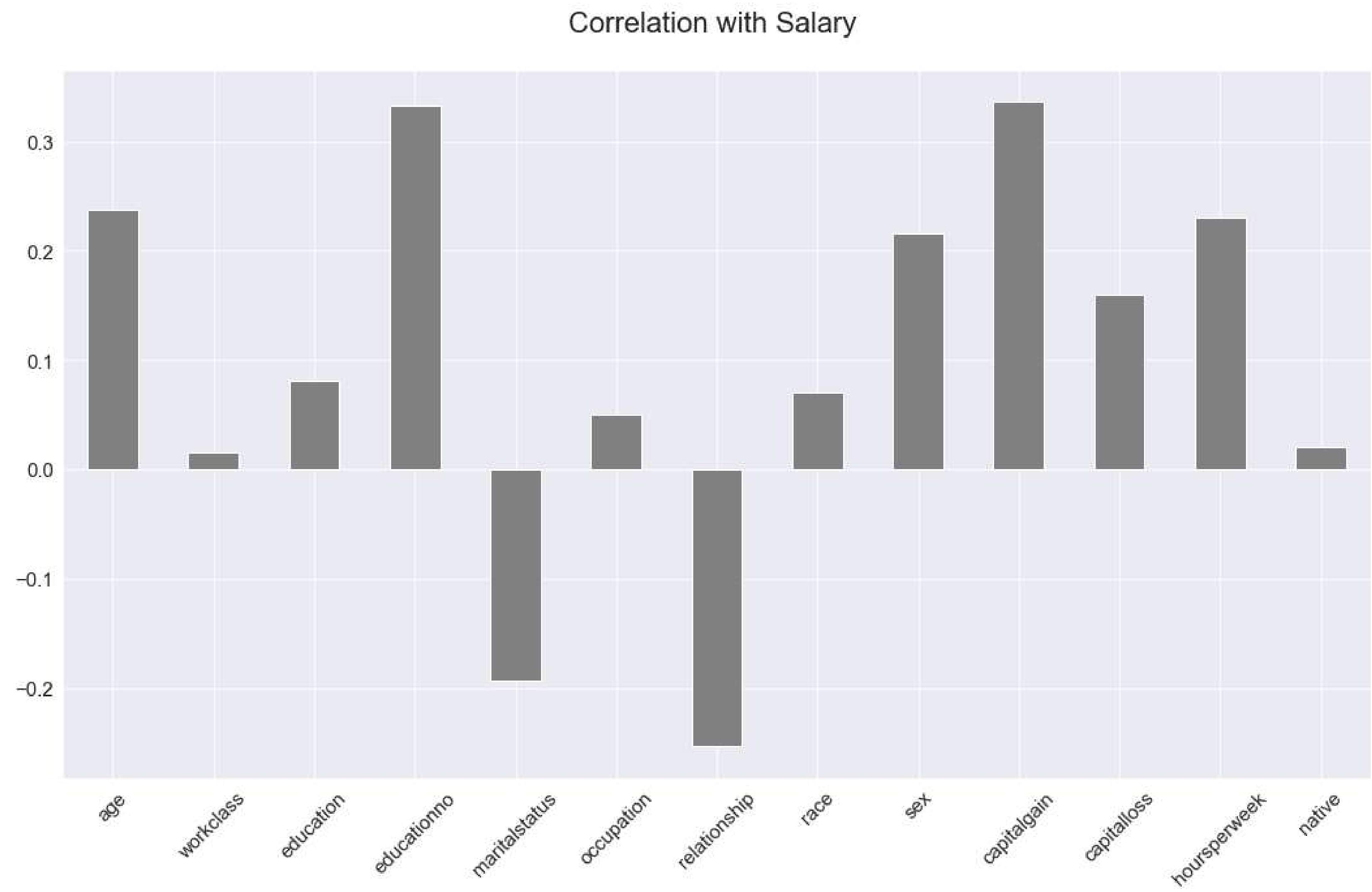
Out[32]:

| | age | workclass | education | educationno | maritalstatus | occupation | relationship | race | sex | capitalgain | capitalloss | hoursperweek |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 22 | 5 | 9 | 12 | 4 | 0 | 1 | 4 | 1 | 24 | 0 | 39 |
| **1** | 33 | 4 | 9 | 12 | 2 | 3 | 0 | 4 | 1 | 0 | 0 | 12 |
| **2** | 21 | 2 | 11 | 8 | 0 | 5 | 1 | 4 | 1 | 0 | 0 | 39 |
| **3** | 36 | 2 | 1 | 6 | 2 | 5 | 0 | 2 | 1 | 0 | 0 | 39 |
| **4** | 11 | 2 | 9 | 12 | 2 | 9 | 5 | 2 | 0 | 0 | 0 | 39 |

In [33]:

```
# correlation woith salary

data2 = data_copy.iloc[:,:-1]

correlations = data2.corrwith(data_copy["Salary"])
correlations = correlations[correlations!=1]
positive_correlations = correlations[correlations >0].sort_values(ascending = False)
negative_correlations =correlations[correlations<0].sort_values(ascending = False)

correlations.plot.bar(
        figsize = (18, 10),
        fontsize = 15,
        color = 'grey',
        rot = 45, grid = True)
plt.title('Correlation with Salary \n',
horizontalalignment="center", fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
```

Out[33]: Text(0.5, 1.0, 'Correlation with Salary \n')



In [34]:
```python
import seaborn as sns
import matplotlib.pyplot as pplt
#correlation matrix
corrmat = data_copy.corr()
f, ax = pplt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True, annot=True);
```


In [40]:
```python
y = train_data['Salary']
X = train_data.drop("Salary", axis=1)
```

In [41]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [44]:
```python
# Preparing a naive bayes model on training data set

from sklearn.naive_bayes import MultinomialNB as MB
from sklearn.naive_bayes import GaussianNB as GB

# Multinomial Naive Bayes
model_gb = MB()
model_gb.fit(X_train, y_train)
```

Out[44]:
```
MultinomialNB()
```

In [46]:

```
predict = model_gb.predict(X_test)
print(classification_report(y_test, predict))

sns.heatmap(confusion_matrix(y_test, predict),annot=True,fmt = "d",linecolor="k",linewidths=3)
```

```
              precision    recall  f1-score   support

           0       0.80      0.94      0.86      7466
           1       0.62      0.31      0.41      2488

    accuracy                           0.78      9954
   macro avg       0.71      0.62      0.64      9954
weighted avg       0.76      0.78      0.75      9954
```

Out[46]: <AxesSubplot:>


In [54]:

```
score_gb = model_gb.score(X_test,y_test)
print('The accuracy of Gaussian Naive Bayes is', score_gb)
```

```
The accuracy of Gaussian Naive Bayes is 0.7799879445449066
```

In [67]:

```
!pip install pipeline
```

```
Collecting pipeline
  Downloading pipeline-0.1.0-py3-none-any.whl (2.6 kB)
Installing collected packages: pipeline
Successfully installed pipeline-0.1.0
```

In [125]:

```
nb_classifier = GB()
skf = StratifiedKFold(n_splits=9)
params_NB = {'var_smoothing': np.logspace(0,-9, num=100),

    #'var_smoothing': [0.00000001, 0.000000001, 0.00000001]
            }
gs_NB = GridSearchCV(nb_classifier,
                 param_grid=params_NB,
                 cv=skf,
                 verbose=10,
                 scoring='accuracy')
gs_NB.fit(X_train, y_train)

gs_NB.best_score_
```

```
[CV]  var_smoothing=3.5111917342151277e-08, score=0.816, total=   0.0s
[CV] var_smoothing=3.5111917342151277e-08 ............................
[CV]  var_smoothing=3.5111917342151277e-08, score=0.810, total=   0.0s
[CV] var_smoothing=3.5111917342151277e-08 ............................
[CV]  var_smoothing=3.5111917342151277e-08, score=0.821, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.809, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.821, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.812, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.817, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.818, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.792, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
[CV] . var_smoothing=2.848035868435799e-08, score=0.816, total=   0.0s
[CV] var_smoothing=2.848035868435799e-08 ............................
```

In [129]:

```
gs_NB.best_params_
```

Out[129]: {'var_smoothing': 5.336699231206313e-06}

In [130]:
```
score_ = gs_NB.score(X_test,y_test)
print('The accuracy of Gaussian Naive Bayes is', score_)
```

```
The accuracy of Gaussian Naive Bayes is 0.8131404460518384
```

In [131]:
```
predict_gb = gs_NB.predict(X_test)
print(classification_report(y_test, predict_gb))

sns.heatmap(confusion_matrix(y_test, predict_gb),annot=True,fmt = "d",linecolor="k",linewidths=3)
```

```
              precision    recall  f1-score   support

           0       0.84      0.93      0.88      7466
           1       0.68      0.47      0.56      2488

    accuracy                           0.81      9954
   macro avg       0.76      0.70      0.72      9954
weighted avg       0.80      0.81      0.80      9954
```

Out[131]: <AxesSubplot:>