```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 from sklearn.datasets.base import Bunch
5 from sklearn.preprocessing import LabelEncoder
6 from sklearn.base import BaseEstimator, TransformerMixin
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprec
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The sklearn.datasets.base moc
  warnings.warn(message, FutureWarning)
```

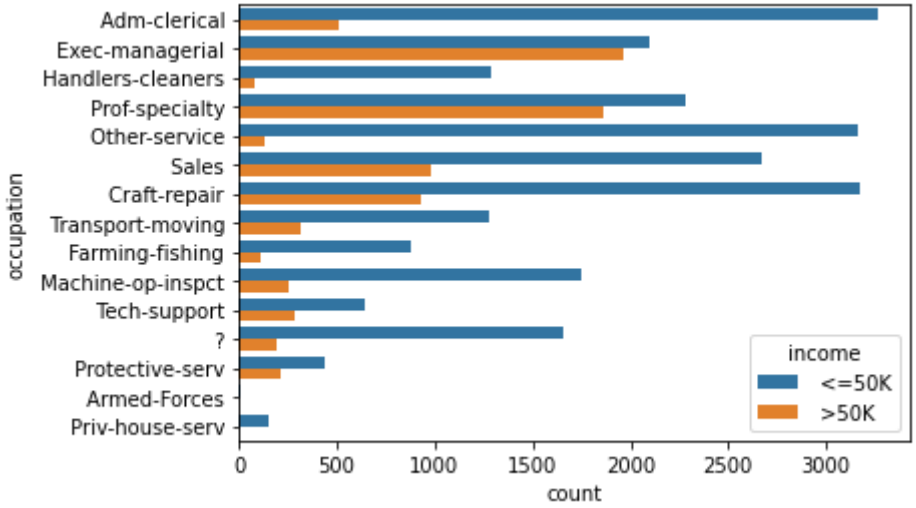
Read in the data and explore it

```
1
2 on-num','marital-status','occupation','relationship','race','sex','capital-gain','capital-loss','hours-per-week','nati
```

```
1 #reading in the data
2 adult_data_train=pd.read_csv('adult.data',names=col_names)
3 adult_data_predict= pd.read_csv('adult.test', names=col_names, skiprows=1)
```

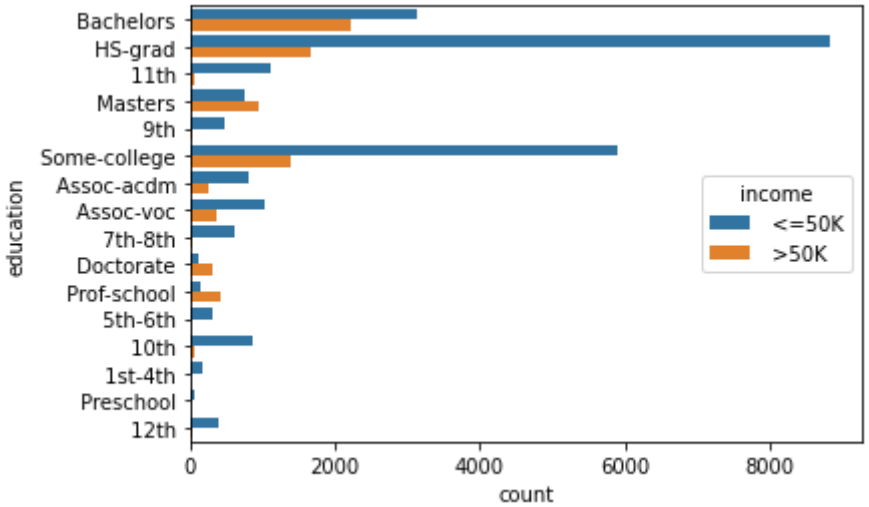
```
1 sns.countplot(y='occupation', hue='income', data=adult_data_train,)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a2c813b00>
```



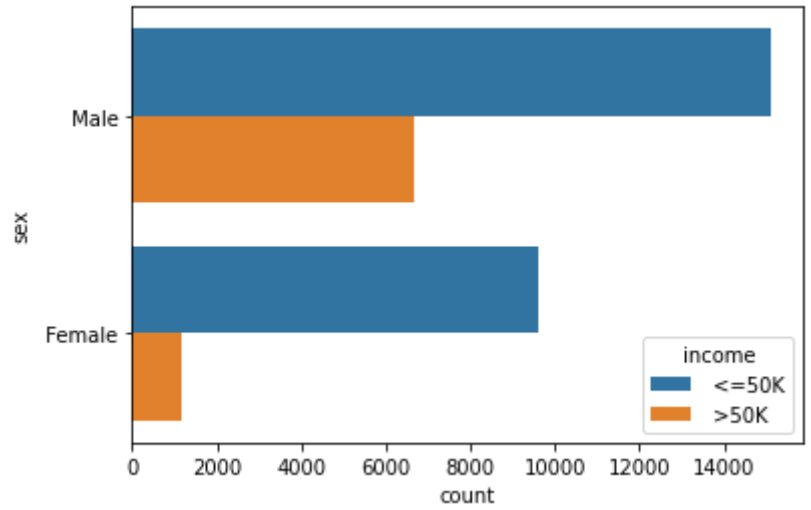

```
1 sns.countplot(y='education', hue='income', data=adult_data_train,)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a2c6c6e80>
```



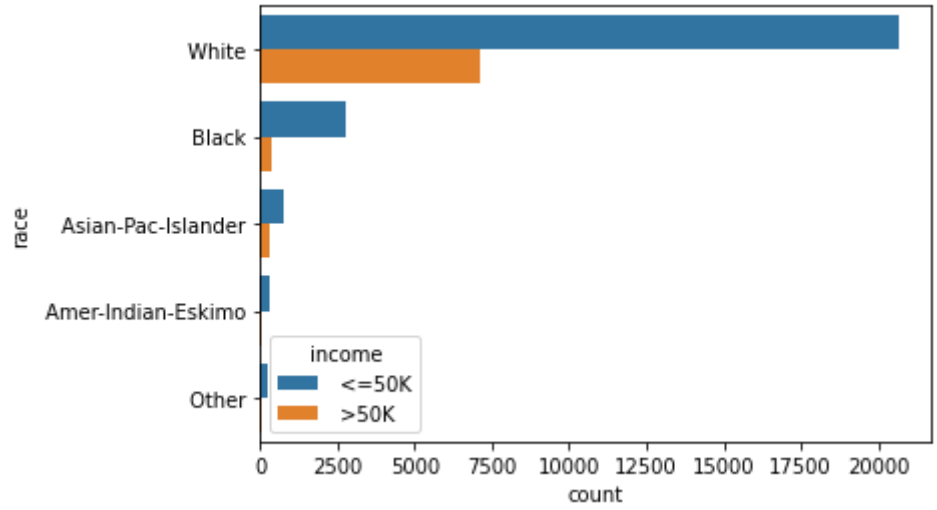

```
1 sns.countplot(y='sex', hue='income', data=adult_data_train,)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f2a2c1ce048>



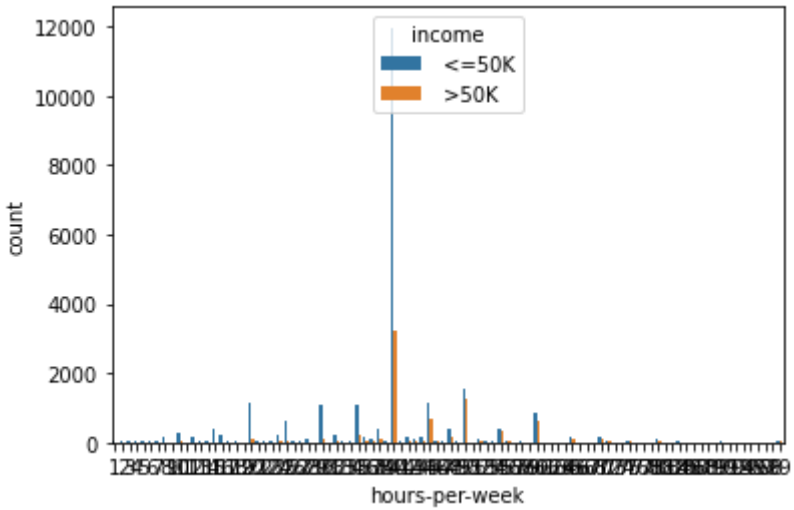

```
1 sns.countplot(y='race', hue='income', data=adult_data_train,)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f2a2c138a90>




```
1 sns.countplot(x='hours-per-week', hue='income', data=adult_data_train,)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a2c0796d8>
```



Preprocessing the data

```
#1. Replace the Nan/Missing data values with mode values
attr_list=list(col_names)
attr_list.remove('income')

for col in attr_list:
    mode=adult_data_train[col].mode()
    adult_data_train[col].fillna(mode)

for col in attr_list:
    mode=adult_data_predict[col].mode()
    adult_data_predict[col].fillna(mode)
```

```
#2. For encoding the categorical data
class EncodeCategorical(BaseEstimator, TransformerMixin):
    """
    Encodes a specified list of columns or all columns if None
```

```
    """
    def __init__(self, columns=None):
        self.columns  = columns
        self.encoders = None
    def fit(self, data, target=None):
        """
        Expects a data frame with named columns to encode.
        """
        # Encode all columns if columns is None
        if self.columns is None:
            self.columns = data.columns
        # Fit a label encoder for each column in the data frame
        self.encoders = {
            column: LabelEncoder().fit(data[column])
            for column in self.columns
        }
        return self
    def transform(self, data):
        """
        Uses the encoders to transform a data frame.
        """
        output = data.copy()
        for column, encoder in self.encoders.items():
            output[column] = encoder.transform(data[column])
        return output

encoder = EncodeCategorical(col_names)
data = encoder.fit_transform(adult_data_train)
data_predict=encoder.fit_transform(adult_data_predict)
```

```
#3. Split into the train and test sets
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data,adult_data_train['income'],test_size=0.2)
```

The KNN Model

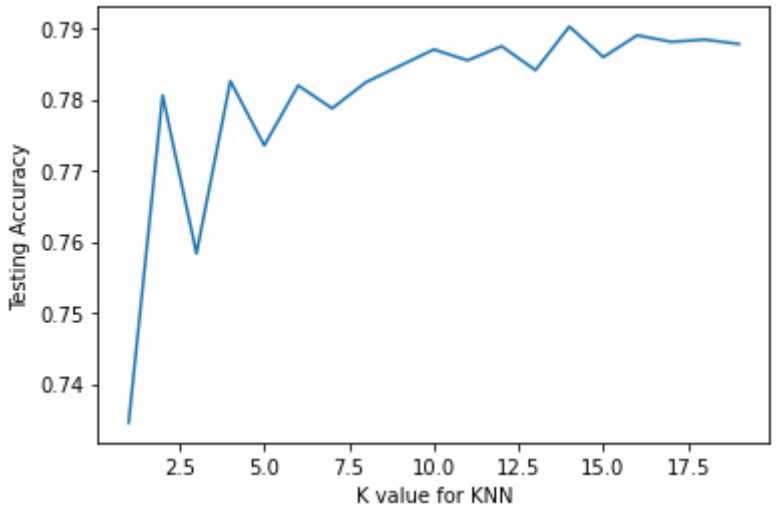
```
#1.Find the optimum value of k
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

k_range=range(1,20)
scores={}
scores_list=[]

for k in k_range:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    y_pred=knn.predict(X_test)
    sc=metrics.accuracy_score(y_test,y_pred)
    scores[k]=sc
    scores_list.append(sc)

plt.plot(k_range,scores_list)
plt.xlabel('K value for KNN')
plt.ylabel('Testing Accuracy')
```

```
Text(0, 0.5, 'Testing Accuracy')
```




```
#Train the knn model with optimum value of k
```

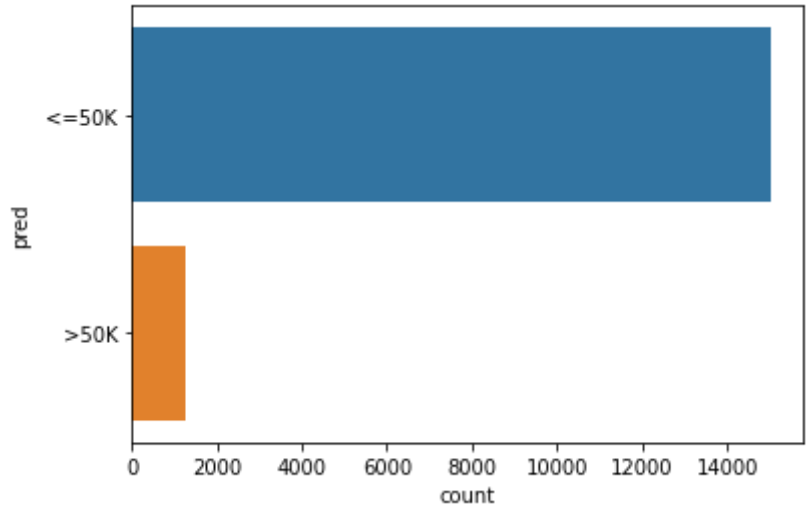
```
2 k_optimum=17
3 knn=KNeighborsClassifier(n_neighbors=k_optimum)
4 knn.fit(X_train,y_train)
5 knn.score(X_test, y_test)
```

```
0.7881160755412252
```

```
1 #predict the value for the test sets
2 prediction=knn.predict(data_predict)
```

```
1 pred=pd.DataFrame(prediction,columns=['pred'])
2 sns.countplot(y='pred',data=pred,)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2a29551e10>
```




```
1
```