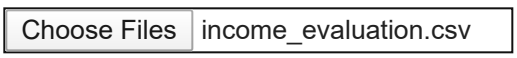
Classification on Income Dataset

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

%matplotlib inline
from sklearn.preprocessing import StandardScaler,LabelEncoder,RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import cohen_kappa_score,accuracy_score,confusion_matrix,recall_score
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
from google.colab import files
import io
import pandas as pd
```

```
pd.set_option('display.max_columns', None)
uploaded = files.upload()
```



- **income_evaluation.csv**(application/vnd.ms-excel) - 3811669 bytes, last modified: 10/20/2019 - 100% don

```
Saving income_evaluation.csv to income_evaluation.csv
```

```
income1 = pd.read_csv("income_evaluation.csv")
```

Data Cleaning

Dealing with Missing Data

```
income1.shape
```

```
(32561, 15)
```

```
income1.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupati
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-cleric
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-manager
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleane
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleane
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specia

```
income1=income1.rename(columns={" workclass": "Workclass"," fnlwgt": "FinalWeight","eductaion
" marital-status": "Marital.Status", " relationship": "Relationship", " race": "Race", " sex"
" hours-per-week": "Hours.Per.Week", " native-country": "Native.Country"," income": "Income.G
```

```
income1=income1.rename(columns={' education': 'Education'})
```

```
income1.head()
```

	age	Workclass	FinalWeight	Education	EducationLevel	Marital.Status	Oc
0	39	State-gov	77516	Bachelors	13	Never-married	Ad
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-m
2	38	Private	215646	HS-grad	9	Divorced	Handlers
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof

```
income1.Workclass.value_counts()
```

```
Private             22696
Self-emp-not-inc     2541
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: Workclass, dtype: int64
```

```
income1.Workclass.fillna(value="Private", inplace = True )
```

```
income1.Occupation.value_counts()
```

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: Occupation, dtype: int64
```

```
income1[income1.Workclass<="Private"]["Occupation"].value_counts()
```

```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: Occupation, dtype: int64
```

```
income1.Occupation.fillna(value="Prof-specialty", inplace=True)
```

```
income1["Native.Country"].value_counts()
```

```
United-States                 29170
Mexico                          643
?                               583
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Guatemala                        64
Japan                            62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
Greece                           29
France                           29
Ecuador                          28
Ireland                          24
Hong                             20
Trinadad&Tobago                  19
Cambodia                         19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Honduras                         13
Hungary                          13
Scotland                         12
Holand-Netherlands                1
Name: Native.Country, dtype: int64
```

```
income1[income1["Native.Country"] <= "United-States"]["Income.Group"].value_counts()
```

```
<=50K    24720
>50K      7841
Name: Income.Group, dtype: int64
```

```
income1[income1["Native.Country"] <= "United-States"]["Income.Group"].value_counts()
```

```
 <=50K    24720
 >50K      7841
Name: Income.Group, dtype: int64
```

```
income1[income1["Native.Country"].isnull()]["Income.Group"].value_counts()
```

```
Series([], Name: Income.Group, dtype: int64)
```

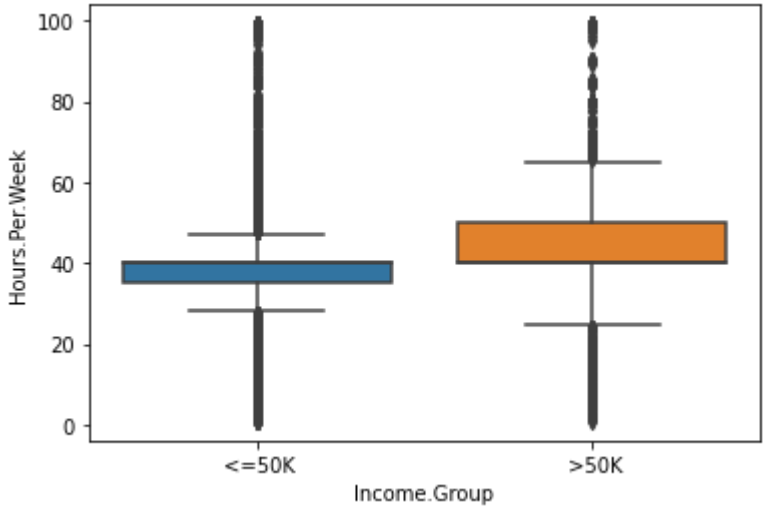
```
income1["Native.Country"].fillna(value="United-States",inplace=True)
```

```
income1["Hours.Per.Week"].value_counts()
```

```
40    15217
50     2819
45     1824
60     1475
35     1297
      ...
92        1
94        1
87        1
74        1
82        1
Name: Hours.Per.Week, Length: 94, dtype: int64
```

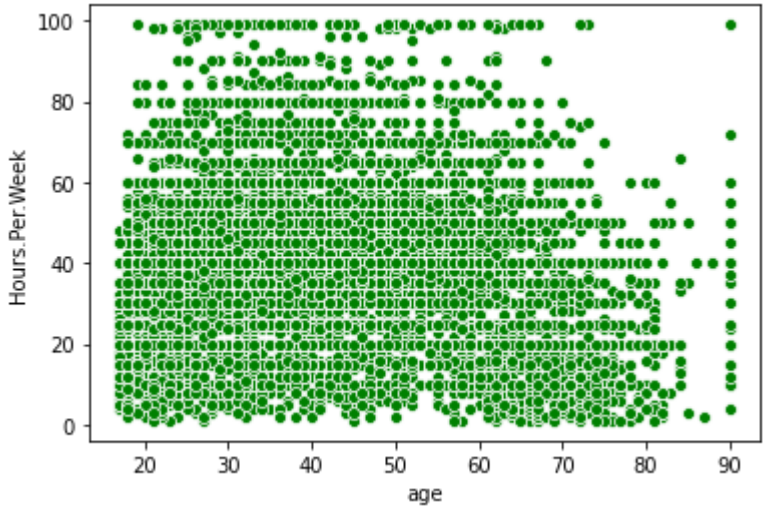
```
sns.boxplot(x="Income.Group" , y = "Hours.Per.Week" , data = income1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe4d1520278>
```




```
sns.scatterplot(x="age" , y = "Hours.Per.Week" , data = income1, color='green')
```

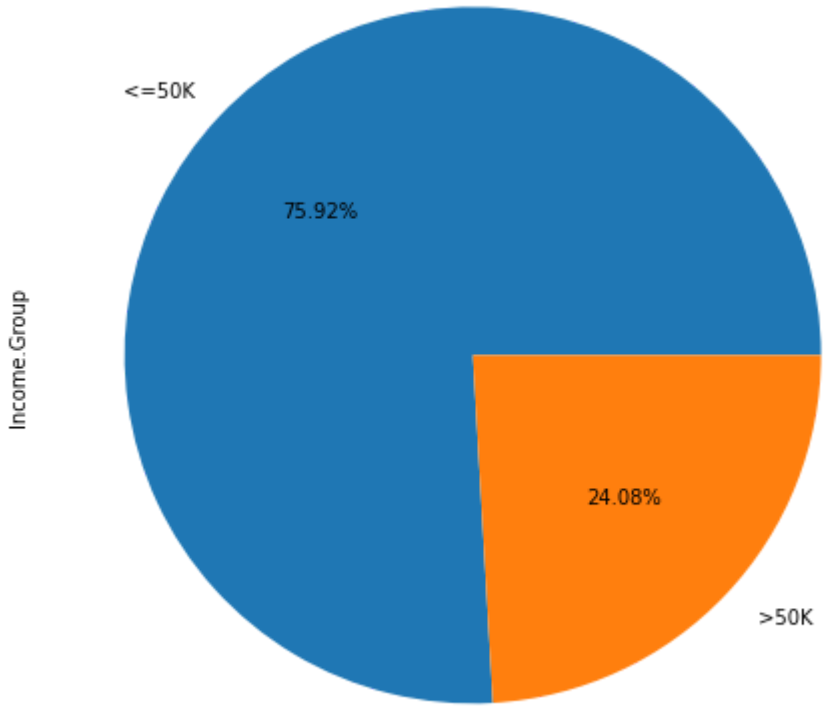
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe4d1520dd8>
```



Data Exploration

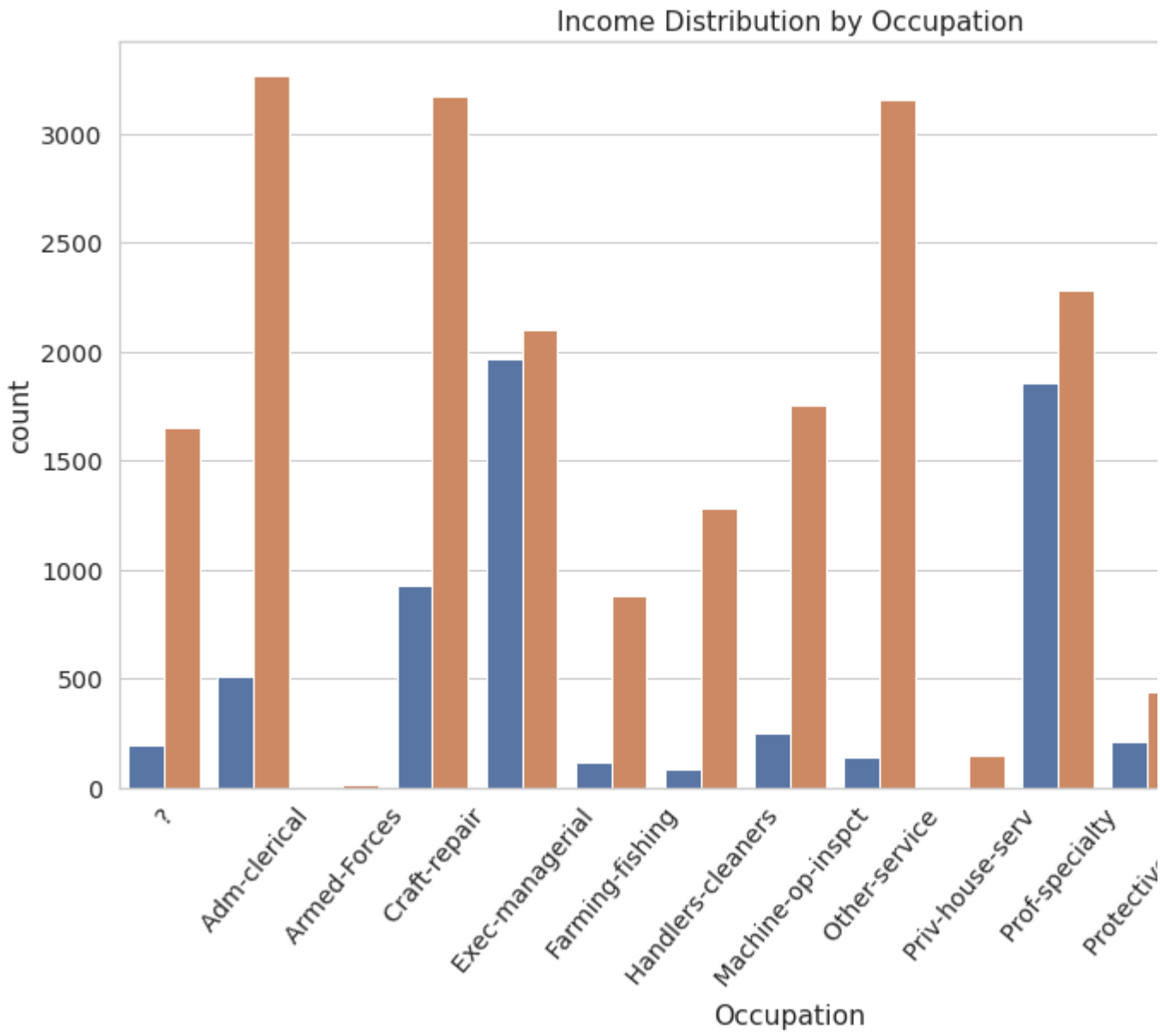
```
income_counts = income1['Income.Group'].value_counts()
income_counts.plot(kind='pie', subplots=True, figsize=(8, 8), autopct='%.2f%%')
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fe4ce6f6240>],
      dtype=object)
```



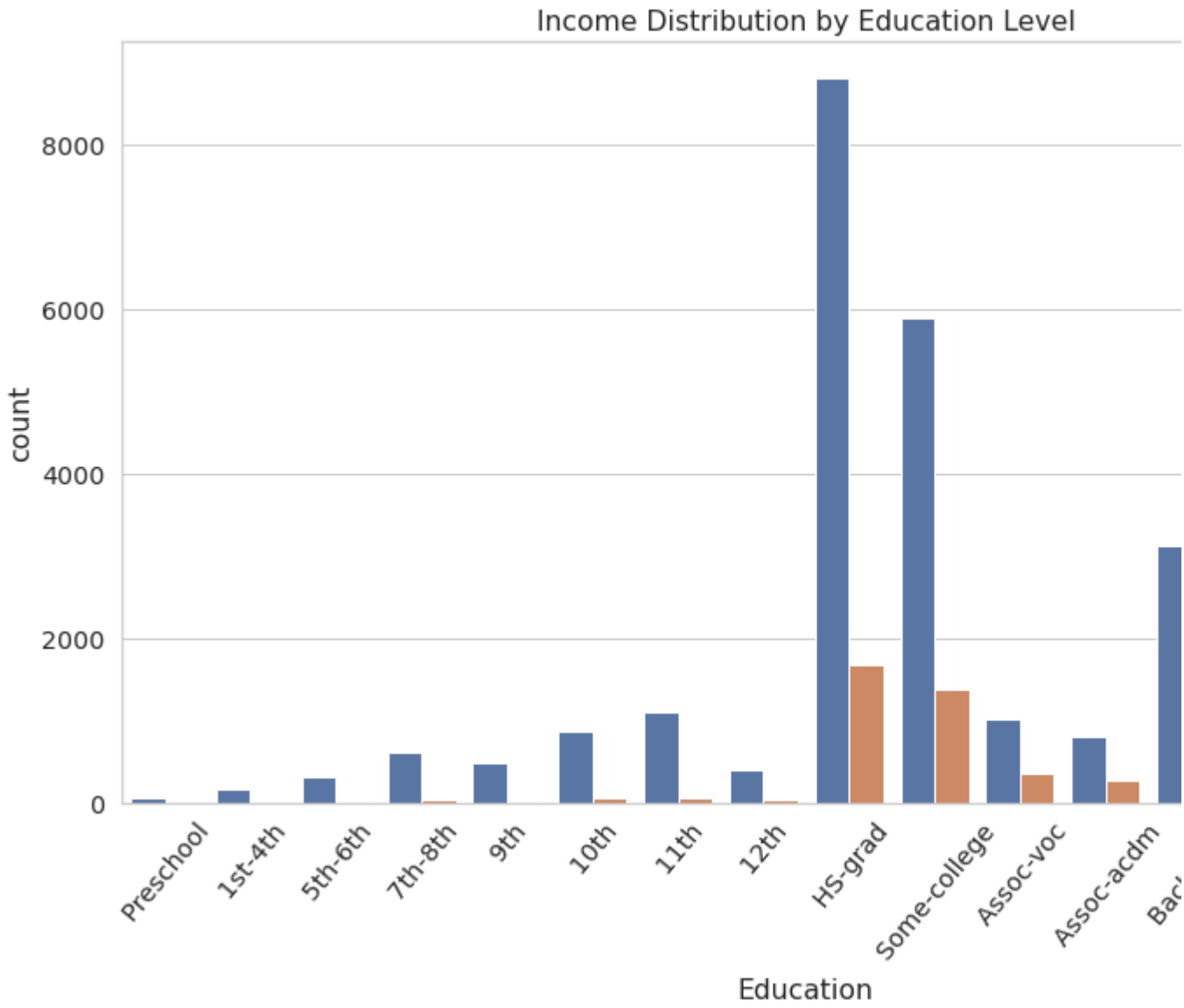

```
#Create Dataframe of Occupation and Income Categories
```

```
#Create Dataframe of Occupation and Income Categories
inc_occ_df = income1[['Occupation','Income.Group']].sort_values(by =['Occupation'])

#Create Seaborn Nested/Multiple Count Plot
sns.set(style="whitegrid", font_scale=1.3)
plt.figure(figsize=(14,8))
cplot = sns.countplot(x="Occupation", hue="Income.Group", data=inc_occ_df)
cplot.set_title('Income Distribution by Occupation')

#Rotate X-Tick Labels
for item in cplot.get_xticklabels():
    item.set_rotation(50)
```




```
#Create Dataframe of Education and Income Categoris
inc_occ_df = income1[['Education','Income.Group','EducationLevel']].sort_values(by =['Educati

#Create Seaborn Nested/Multiple Count Plot
sns.set(style="whitegrid", font_scale=1.3)
```

```
plt.figure(figsize=(14,8))
cplot = sns.countplot(x="Education", hue="Income.Group", data=inc_occ_df)
cplot.set_title('Income Distribution by Education Level')


#Rotate X-Tick Labels
for item in cplot.get_xticklabels():
    item.set_rotation(50)
```

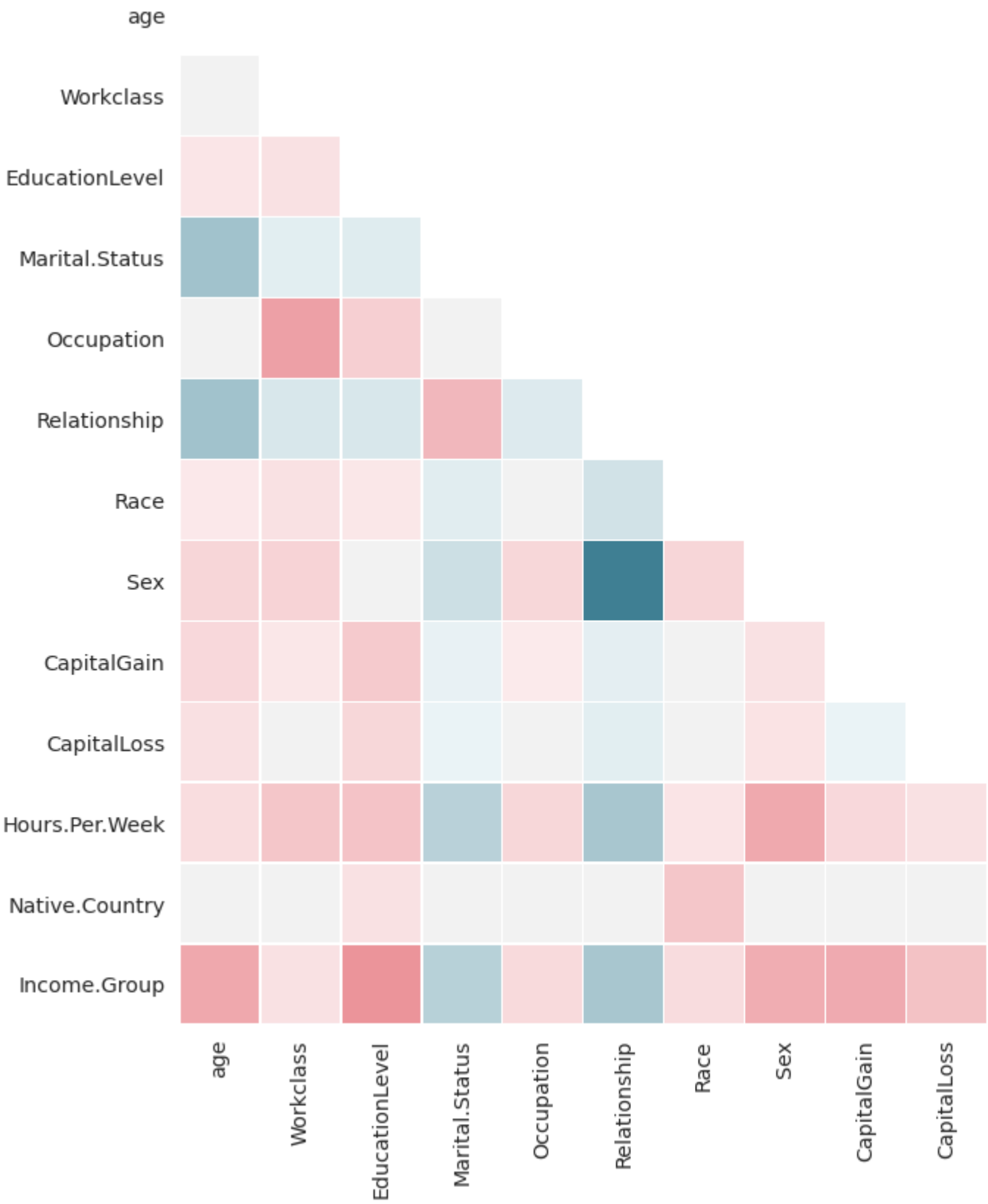



```
#Label Encoder Function
def encode_data (df):
    # Cycle Through Each Column to Transform
    for column in df.columns:
        #Only Encode if Object Type
        if df[column].dtypes == object:
            df[column]=LabelEncoder().fit_transform(df[column])
    return df


training encoded = encode data(income1)
```

```
training_encoded['Income.Group'] = income1['Income.Group'].astype('object')
```

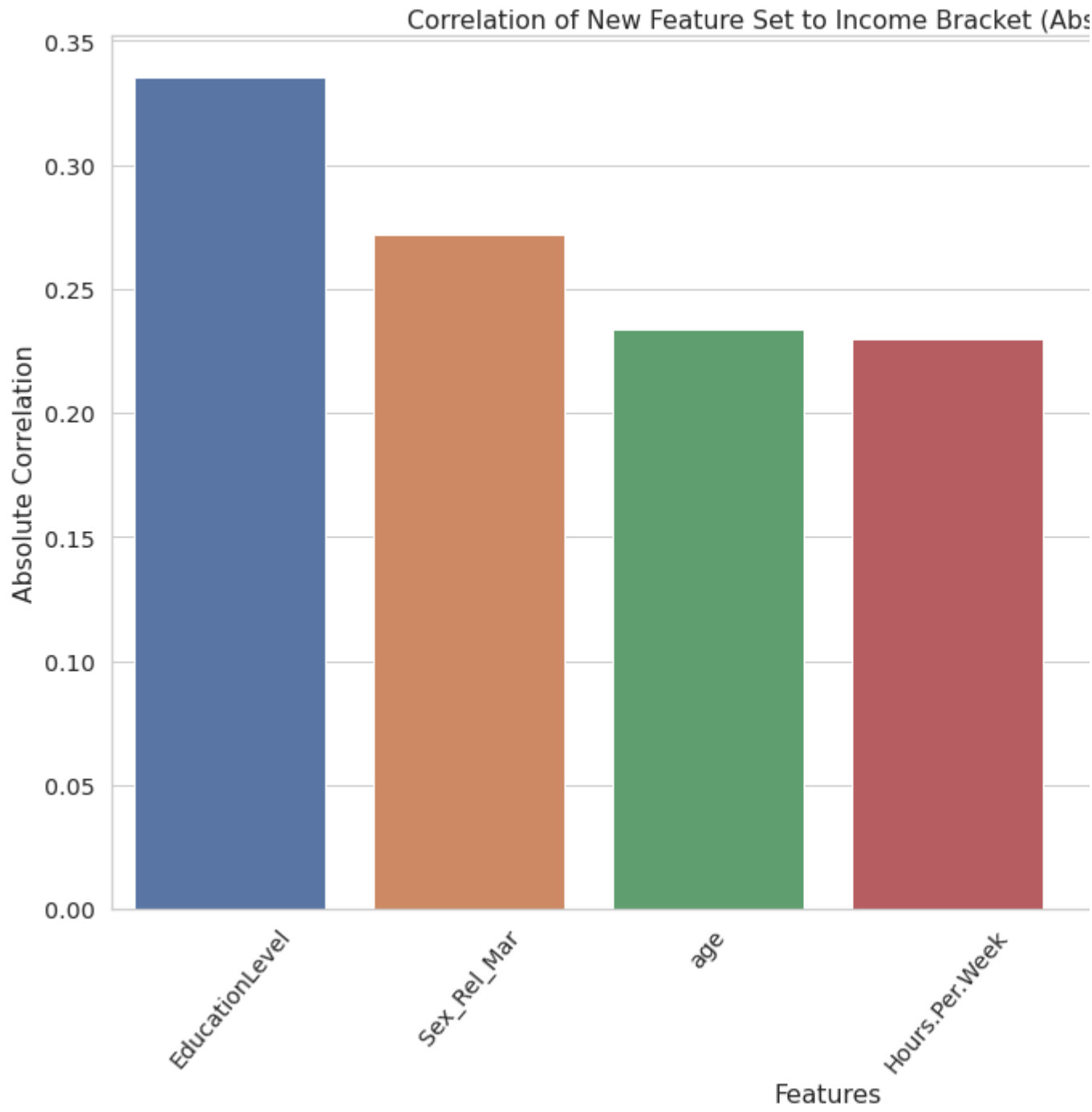
```
# Compute the correlation matrix
training_encoded['Income.Group'] = income1['Income.Group'].astype('object')
training_encoded2 = encode_data(training_encoded)
features = training_encoded2[['age','Workclass','EducationLevel','Marital.Status','Occupation
corr = features.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(16, 14))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe4ccc34240>
```



FS

```
#Remove Insignificant Features
training_encoded_short = training_encoded.drop(['Race','Native.Country','Workclass','Income.G
                                                ,'Sex','Relationship','FinalWeight','Marital.
                                                ,'CapitalGain','CapitalLoss','Education' ], a
#Create New Features
training_encoded_short['Sex_Rel_Mar'] = pd.DataFrame(training_encoded["Sex"] + training_encod
                                                     + training_encoded['Marital.Status'])
training_encoded_short['Net_Capital'] = pd.DataFrame(training_encoded["CapitalGain"] - traini



training_encoded_short['Income.Group'] = income1['Income.Group'].astype('object')
training_encoded_short2 = encode_data(training_encoded_short)
corr2 = training_encoded_short2.corr()

#Plot Absolute Correlation Values for Each Feature
abs_corr2 = abs(corr2.iloc[0:6,6:7]).sort_values(by=['Income.Group'], ascending=False)

#Create Bar Plot of Feature Correlation Values
sns.set(style="whitegrid", font_scale=1.3)
plt.figure(figsize=(16,10))
bplot = sns.barplot(x=abs_corr2.index, y = 'Income.Group',  data=abs_corr2)
bplot.set(xlabel='Features', ylabel='Absolute Correlation')
bplot.set_title('Correlation of New Feature Set to Income Bracket (Absolute Value)')

#Rotate X-Tick Labels
for item in bplot.get_xticklabels():
    item.set_rotation(50)
```



Model Implementation

```
ohe=pd.get_dummies(income1.drop(["Income.Group"] , axis = 1 ))

lb = LabelEncoder()

y = pd.DataFrame(lb.fit_transform(income1["Income.Group"]))

sc=StandardScaler()
```

```
x = pd.DataFrame(sc.fit_transform(ohe))
```

Split dataframe

```
xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=0.25)
```

Logistic Regression

```
lg=LogisticRegression()
```

```
model_lg=lg.fit(xtrain,ytrain).predict(xtest)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWa
  y = column_or_1d(y, warn=True)
```

```
print("Accuracy score is %.2f for logistic regression" % (accuracy_score(ytest,model_lg)))
```

```
Accuracy score is 0.83 for logistic regression
```

KNeighbors Classifier

```
knn = KNeighborsClassifier()
```

```
model_kn=knn.fit(xtrain,ytrain).predict(xtest)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: DataConversionWarning: A
  """Entry point for launching an IPython kernel.
```

```
print("Accuracy score is %.2f for KNN" % (accuracy_score(ytest,model_kn)))
```

```
Accuracy score is 0.82 for KNN
```

Naive Bayes Classifier

```
nb= GaussianNB()
```

```
model_nb=nb.fit(xtrain,ytrain).predict(xtest)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/naive_bayes.py:206: DataConversionWarning
  y = column_or_1d(y, warn=True)
```

```
print("Accuracy score is %.2f for Naive Bayes" % (accuracy_score(ytest,model_nb)))
```

```
Accuracy score is 0.80 for Naive Bayes
```

```
cohen_kappa_score(ytest,model_nb)
```

```
0.3529365302036297
```

Decision Tree Classifier

```
dt=DecisionTreeClassifier()
```

```
model_dt=dt.fit(xtrain,ytrain).predict(xtest)
```

```
print("Accuracy score is %.2f for Decision Tree Classfier" % (accuracy_score(ytest,model_dt))
```

```
Accuracy score is 0.80 for Decision Tree Classfier
```

```
cohen_kappa_score(ytest,model_dt)
```

```
0.4677167798831847
```

Support Vector Machine(SVM)

```
from sklearn.svm import SVC
```