Project Milestone for CSE578 (Appendix)

Yu Tung Lin

Arizona State University
ylin364@asu.edu

Appendix

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

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier

column_names = ["age", "workclass", "fnlwgt", "education", "education-num",
    "marital-status", "occupation", "relationship", "race", "sex",
    "capital-gain", "capital-loss", "hours-per-week", "native-country",
    "income"]
dff = pd.read_csv("/content/drive/MyDrive/ CSE578project/adult.data", names =
    column_names, header=None)
df = dff.copy()

df.loc[df.income == ' >50K', 'income'] = int(1)
df.loc[df.income == ' <=50K', 'income'] = int(0)
df['income'] = df['income'].astype('int')

df.loc[df.sex == ' Male', 'sex'] = int(1)
df.loc[df.sex == ' Female', 'sex'] = int(0)
df['sex'] = df['sex'].astype('int')

sns.heatmap(df.corr(), annot = True, fmt = '.2f', cmap = 'coolwarm')

g = sns.FacetGrid(df, col='income')
g = g.map(sns.distplot, "age")

g = sns.FacetGrid(df, col='income')
```

```
g = g.map(sns.distplot, "education-num")

g = sns.FacetGrid(df, col='income')
g = g.map(sns.distplot, "hours-per-week")

fig = plt.figure(figsize=(15,8))
sns.stripplot(x="sex", y="age", hue="income", size=1.4, data=df, dodge=True)
plt.show()

fig = plt.figure(figsize=(14,5))
df.loc[df.workclass == ' ?', 'workclass'] = 'Private'
ax2 = sns.countplot(data= df, x='workclass', hue='income')
ax2.set_title("Income count by work class (by income group)", loc='center',
   fontweight='bold', fontsize=18)
ax2.set_xlabel("Work class")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

fig = plt.figure(figsize=(19,5))
ax2 = sns.countplot(data= df, x='education', hue='income')
ax2.set_title("Income count by education (by income group)", loc='center',
   fontweight='bold', fontsize=18)
ax2.set_xlabel("education")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

graph = sns.PairGrid(df, hue ='income')
graph = graph.map_offdiag(plt.scatter)
graph = graph.add_legend()

fig = plt.figure(figsize=(19,5))
ax2 = sns.countplot(data= df, x='education-num', hue='income')
ax2.set_title("Income count by education-num (by income group)",
   loc='center', fontweight='bold', fontsize=18)
ax2.set_xlabel("education-num")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

df['education-num'].replace([1, 2, 3, 4, 5, 6, 7, 8], 8, inplace = True)
df['education-num'].replace([15, 16], 15, inplace = True)
df['education-num'].value_counts()

fig = plt.figure(figsize=(19,5))
ax2 = sns.countplot(data= df, x='marital-status', hue='income')
ax2.set_title("Income count by marital-status (by income group)",
   loc='center', fontweight='bold', fontsize=18)
ax2.set_xlabel("marital-status")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

df['with_spouse'] = 0
df.loc[df['marital-status'] == ' Married-civ-spouse', 'with_spouse'] = 1
df.loc[df['marital-status'] == ' Married-AF-spouse', 'with_spouse'] = 1
fig = plt.figure(figsize=(19,5))
```

```
ax2 = sns.countplot(data= df, x='with_spouse', hue='income')
ax2.set_title("Income count by with_spouse", loc='center', fontweight='bold',
    fontsize=18)
ax2.set_xlabel("with_spouse")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

fig = plt.figure(figsize=(25,5))
df.loc[(df['occupation'] == ' ?') & (df['workclass'] == ' Never-worked'),
    'occupation'] = 'No-work'
df.loc[(df['occupation'] == ' ?') & (df['education-num'] >= 12),
    'occupation'] = ' Prof-specialty'
df.loc[df.occupation == ' ?', 'occupation'] = ' Other-service'
ax2 = sns.countplot(data= df, x='occupation', hue='income')
ax2.set_title("Income count by occupation (by income group)", loc='center',
    fontweight='bold', fontsize=18)
ax2.set_xlabel("occupation")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

fig = plt.figure(figsize=(25,5))
df.loc[(df['relationship'] == ' Own-child') & (df['sex'] == ' Male'),
    'relationship'] = 'Father'
df.loc[(df['relationship'] == ' Own-child') & (df['sex'] == ' Female'),
    'relationship'] = 'Mother'
df.loc[(df['relationship'] == ' Unmarried') & (df['sex'] == ' Male'),
    'relationship'] = 'Unmarried-Male'
df.loc[(df['relationship'] == ' Unmarried') & (df['sex'] == ' Female'),
    'relationship'] = 'Unmarried-Female'
ax2 = sns.countplot(data= df, x='relationship', hue='income')
ax2.set_title("Income count by relationship (by income group)", loc='center',
    fontweight='bold', fontsize=18)
ax2.set_xlabel("relationship")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

df['alone'] = 0
df.loc[df['relationship'] == ' Not-in-family', 'alone'] = 1
df.loc[df['relationship'] == ' Unmarried', 'alone'] = 1
fig = plt.figure(figsize=(19,5))
ax2 = sns.countplot(data= df, x='alone', hue='income')
ax2.set_title("Income count by alone", loc='center', fontweight='bold',
    fontsize=18)
ax2.set_xlabel("alone")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")


fig = plt.figure(figsize=(25,5))
ax2 = sns.countplot(data= df, x='race', hue='income')
ax2.set_title("Income count by race (by income group)", loc='center',
    fontweight='bold', fontsize=18)
ax2.set_xlabel("race")
ax2.set_ylabel(" ")
```

```
ax2.legend(loc="upper right")

fig = plt.figure(figsize=(25,5))
ax2 = sns.countplot(data= df, x='sex', hue='income')
ax2.set_title("Income count by sex (by income group)", loc='center',
    fontweight='bold', fontsize=18)
ax2.set_xlabel("sex")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

df['capital'] = 0
df['capital'] = df['capital-gain'] - df['capital-loss']
fig, axes = plt.subplots(1, 3, figsize=(10, 10), sharey=True)
fig.suptitle('Capital vs Income')
sns.boxplot(ax=axes[0], x='income', y='capital-gain', data=df)
axes[0].set_title("capital gain")
sns.boxplot(ax=axes[1], x='income', y='capital-loss', data=df)
axes[1].set_title("capital loss")
sns.boxplot(ax=axes[2], x='income', y='capital', data=df)
axes[2].set_title("gain - loss")

df['with_capital'] = 'major'
df.loc[(df['capital'] <= 5000) & (df['capital'] >= -5000), 'with_capital'] =
    'minor'
df.loc[df['capital'] == 0, 'with_capital'] = 'none'

ax2 = sns.countplot(data= df, x='with_capital', hue='income')
ax2.set_title("Income count by with_capital (by income group)", loc='center',
    fontweight='bold', fontsize=18)
ax2.set_xlabel("with_capital")
ax2.set_ylabel(" ")

fig = plt.figure(figsize=(40,7))
ax2 = sns.countplot(data= df, x='hours-per-week', hue='income')
ax2.set_title("Income count by hours-per-week (by income group)",
    loc='center', fontweight='bold', fontsize=18)
ax2.set_xlabel("hours-per-week")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

df['work-time'] = pd.cut(df['hours-per-week'], bins = [0, 30, 41, 100],
    labels = ['LesserHours', 'NormalHours', 'ExtraHours'])
sns.countplot(x = 'income', hue = 'work-time', data = df)

fig = plt.figure(figsize=(70,8))
ax2 = sns.countplot(data= df, x='native-country', hue='income')
ax2.set_title("Income count by native-country (by income group)",
    loc='center', fontweight='bold', fontsize=18)
ax2.set_xlabel("native-country")
ax2.set_ylabel(" ")
ax2.legend(loc="upper right")

a=set(df['occupation'].values.tolist())
print(a)
```

```
a = list(a)
fig, axes = plt.subplots(len(a), 1, figsize=(10, 40), sharex=True)
for i in range(len(a)):
  df_some_rows =  df[df['occupation'] == a[i]]
  x=df_some_rows["education-num"]
  ax = sns.boxplot(ax=axes[i], x=x,  y= df_some_rows['workclass'])
  ax.set(xlabel=None)
  axes[i].set_title(a[i])

a=set(df['occupation'].values.tolist())
print(a)
a = list(a)
fig, axes = plt.subplots(len(a), 1, figsize=(10, 40), sharex=True)
for i in range(len(a)):
  df_some_rows =  df[df['occupation'] == a[i]]
  x=df_some_rows["hours-per-week"]
  ax = sns.boxplot(ax=axes[i], x=x,  y= df_some_rows['workclass'])
  ax.set(xlabel=None)
  axes[i].set_title(a[i])

ax2 = sns.countplot(data= df, x='native-country', hue='race')
ax2.set_title("Income count by native-country (by race group)", loc='center',
   fontweight='bold', fontsize=18)
ax2.set_xlabel("native-country")
ax2.set_ylabel("count")
ax2.legend(loc="upper right")

data = df.drop(columns='education', axis=1)
data['work-time'] = df['work-time'].astype('object')
X = data.drop('income', axis=1)
y = data['income']
cat_cols = X.select_dtypes(include=['object']).columns
encoder = LabelEncoder()
for col in cat_cols:
    X[col] = encoder.fit_transform(X[col])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
   random_state=42)

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)
acc_log_1 = round(logreg.score(X_train, y_train) * 100, 2)
acc_log_2 = round(logreg.score(X_test, y_test) * 100, 2)
acc_log_1, acc_log_2

svc = SVC()
svc.fit(X_train, y_train)
Y_pred = svc.predict(X_test)
acc_svc_1 = round(svc.score(X_train, y_train) * 100, 2)
acc_svc_2 = round(svc.score(X_test, y_test) * 100, 2)
acc_svc_1, acc_svc_2

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
```

```
Y_pred = knn.predict(X_test)
acc_knn_1 = round(knn.score(X_train, y_train) * 100, 2)
acc_knn_2 = round(knn.score(X_test, y_test) * 100, 2)
acc_knn_1, acc_knn_2

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree_1 = round(decision_tree.score(X_train, y_train) * 100, 2)
acc_decision_tree_2 = round(decision_tree.score(X_test, y_test) * 100, 2)
acc_decision_tree_1, acc_decision_tree_2

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, y_train)
acc_random_forest_1 = round(random_forest.score(X_train, y_train) * 100, 2)
acc_random_forest_2 = round(random_forest.score(X_test, y_test) * 100, 2)
acc_random_forest_1, acc_random_forest_2

gbc = GradientBoostingClassifier(n_estimators=2000, learning_rate=0.01,
     max_depth=3, verbose=0, random_state=1)
gbc.fit(X_train, y_train)
acc_gbc_1 = round(gbc.score(X_train, y_train) * 100, 2)
acc_gbc_2 = round(gbc.score(X_test, y_test)*100, 2)
acc_gbc_1, acc_gbc_2

models = pd.DataFrame({ 'Model': ['Support Vector Machines', 'KNN', 'Logistic
   Regression', 'Random Forest', 'Decision Tree', 'GradientBoosting'],
   'Train_Score': [acc_svc_1, acc_knn_1, acc_log_1, acc_random_forest_1,
   acc_decision_tree_1, acc_gbc_1], 'Test_Score': [acc_svc_2, acc_knn_2,
   acc_log_2, acc_random_forest_2, acc_decision_tree_2, acc_gbc_2]})
models
```

Figures

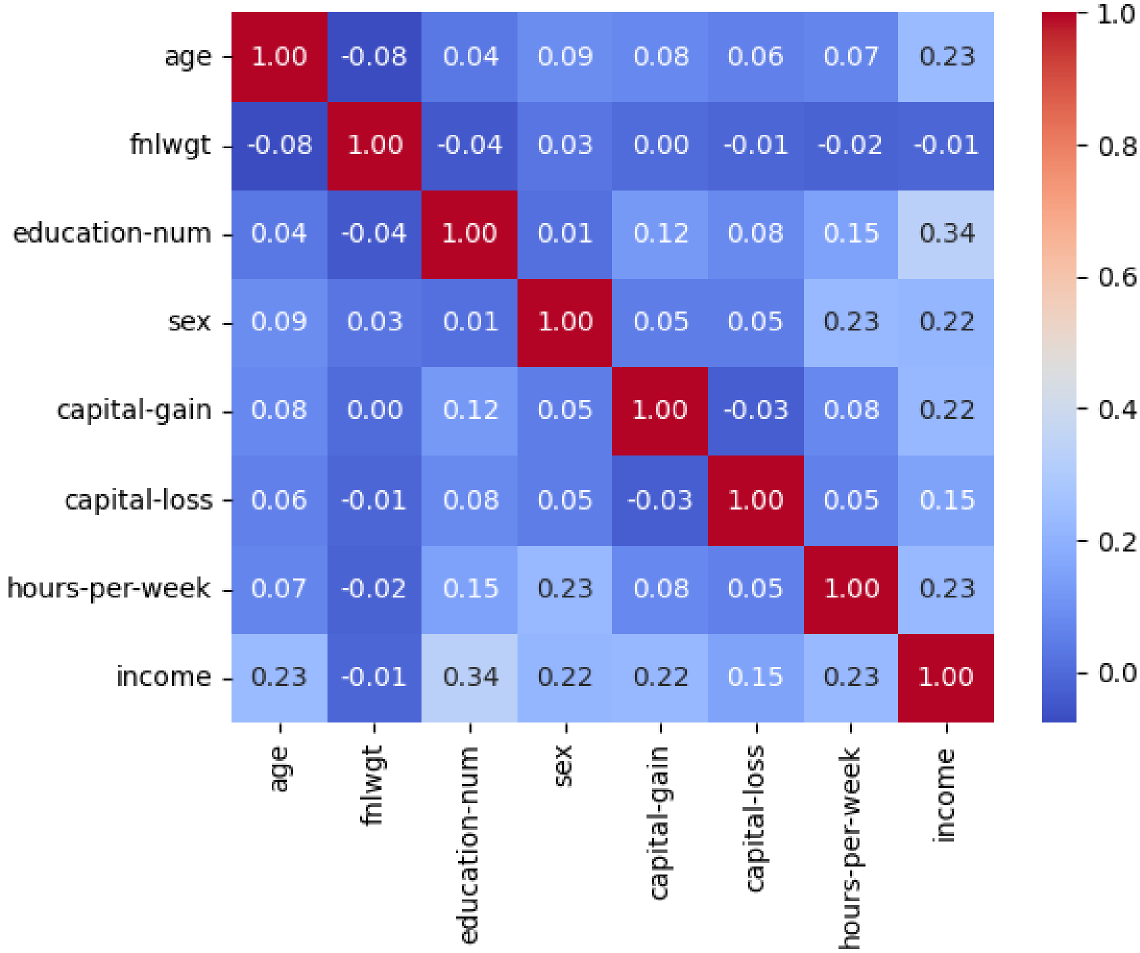

Figure 1: heat map

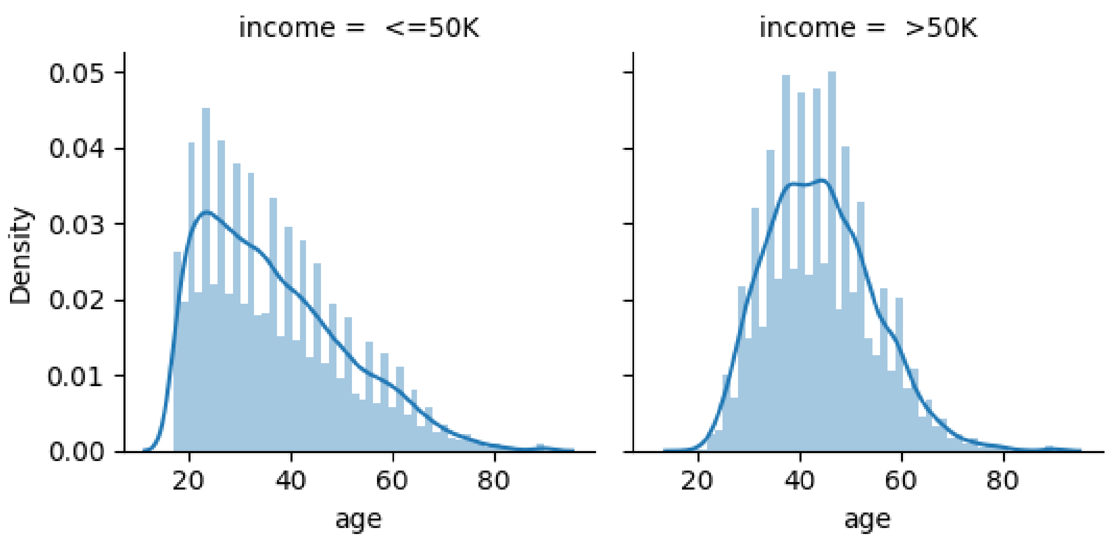

Figure 2: relation between the age and income

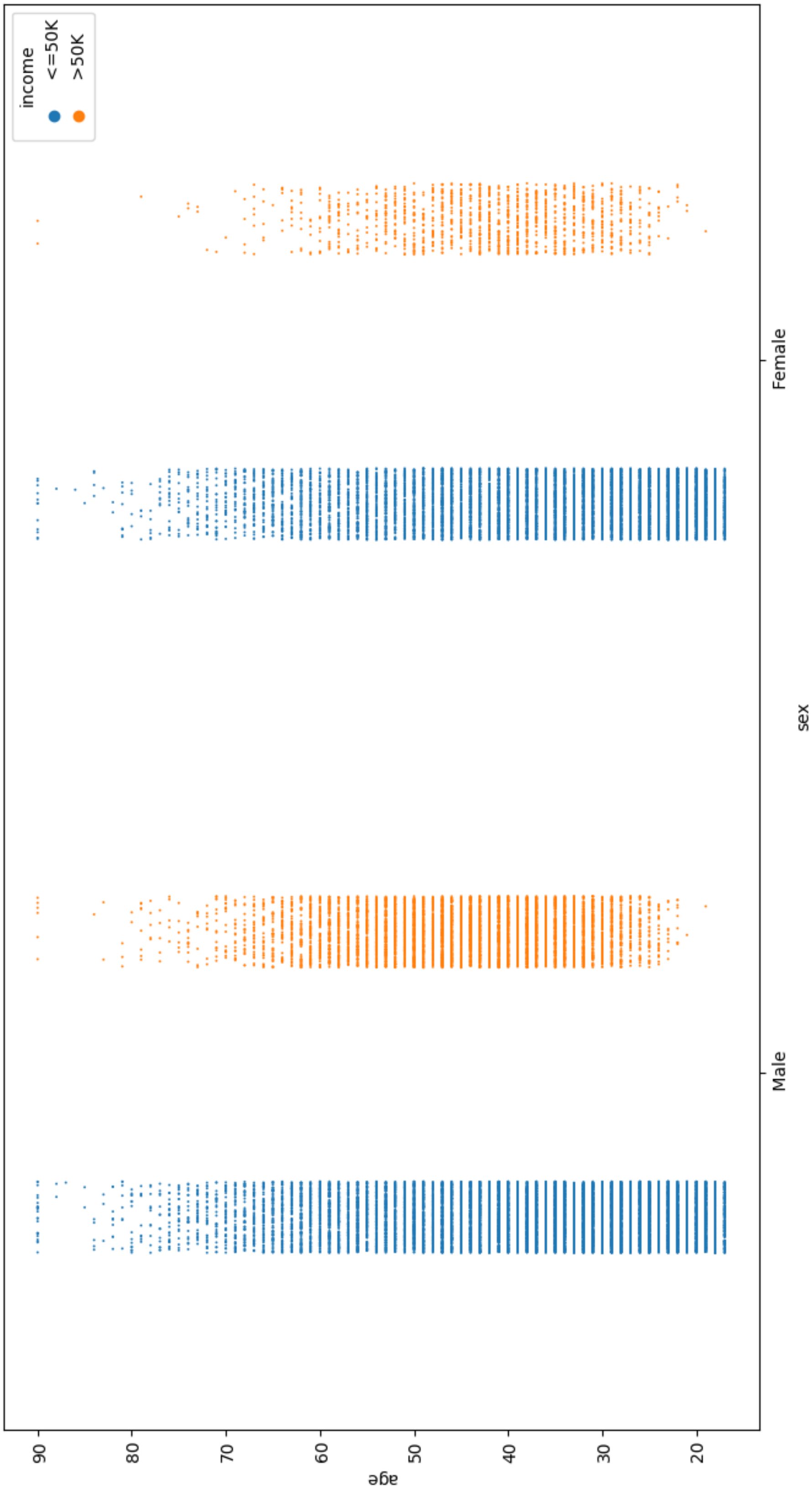

Figure 3: relation between the age, sex, and income

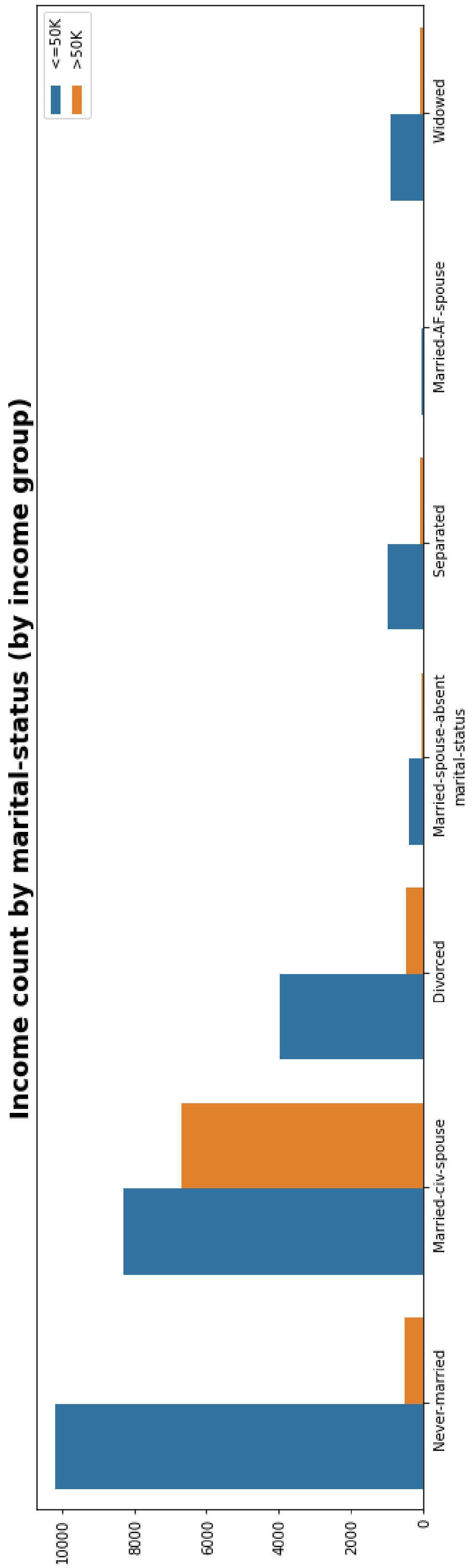

Figure 4: relation between marital status and income

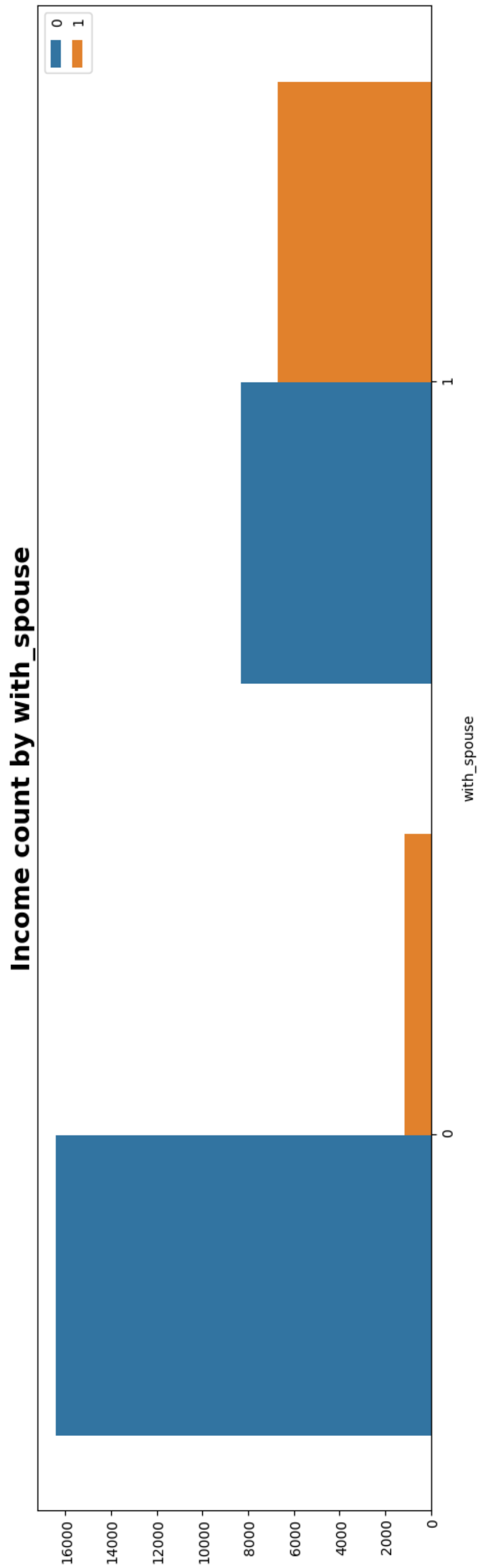

Figure 5: relation between with-spouse and income

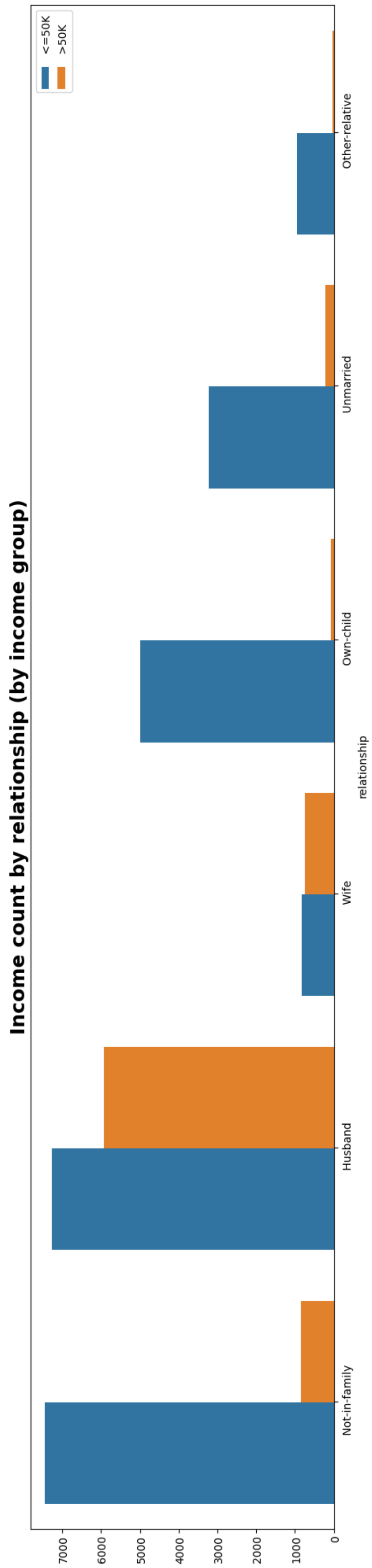

Figure 6: relation between relationship and income

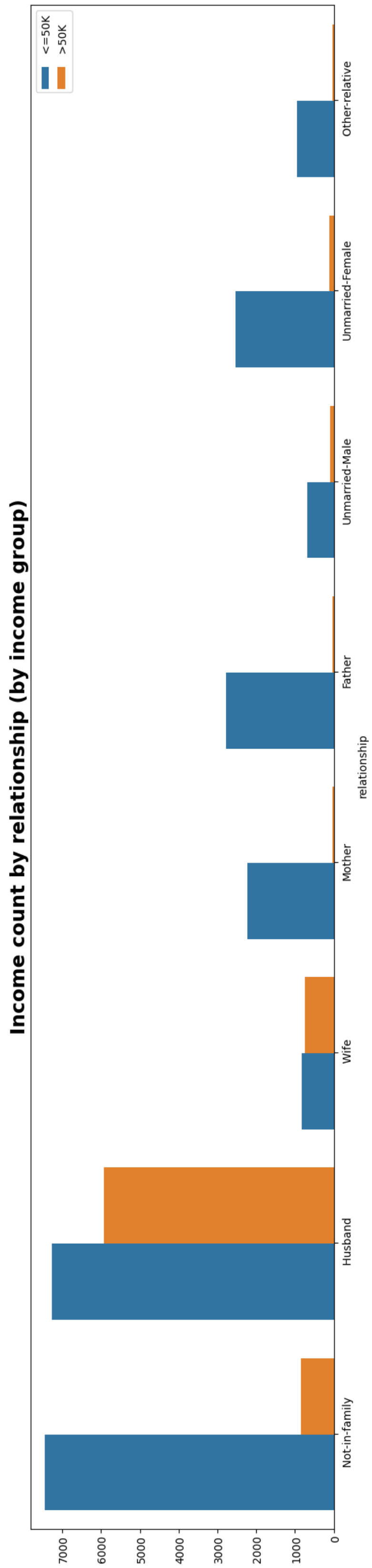

Figure 7: relation between relationship and income

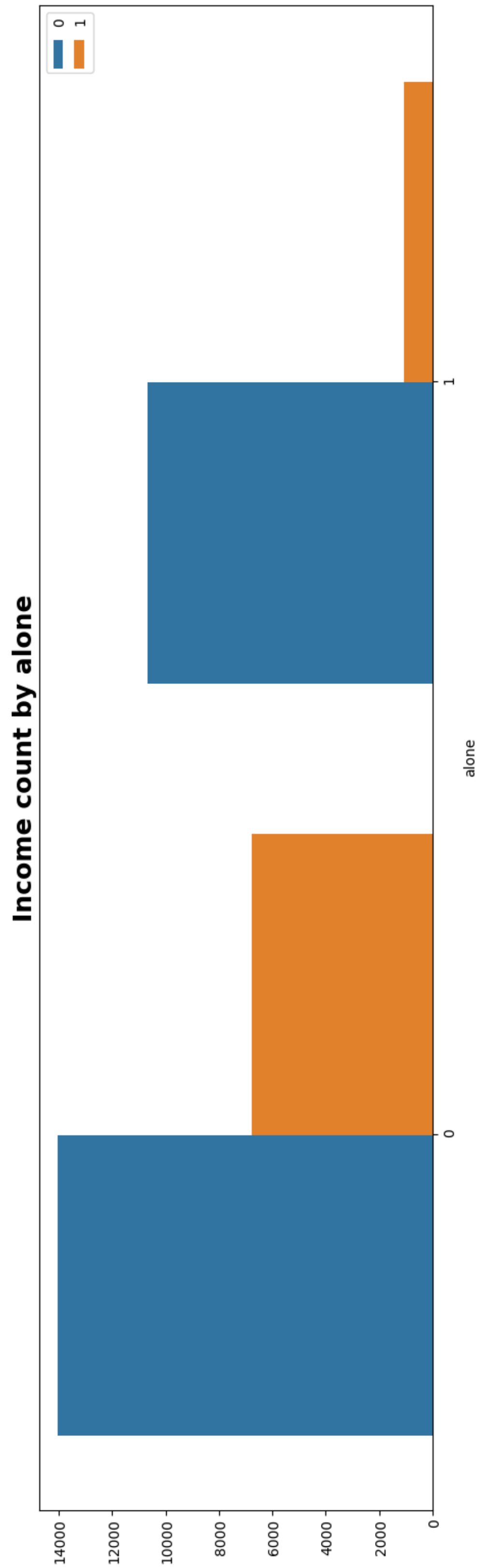

Figure 8: relation between alone and income

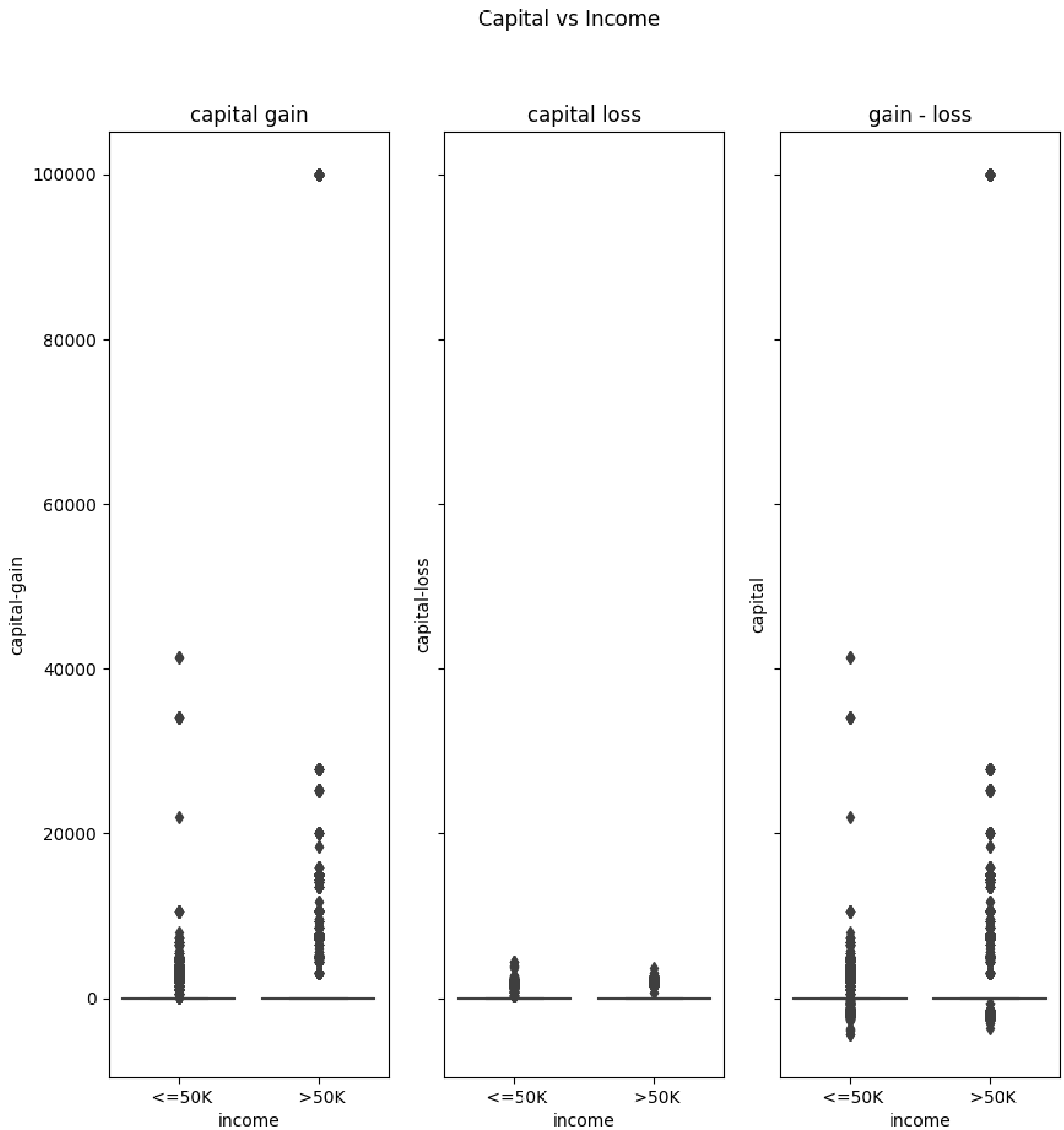

Figure 9: relation between capital and income

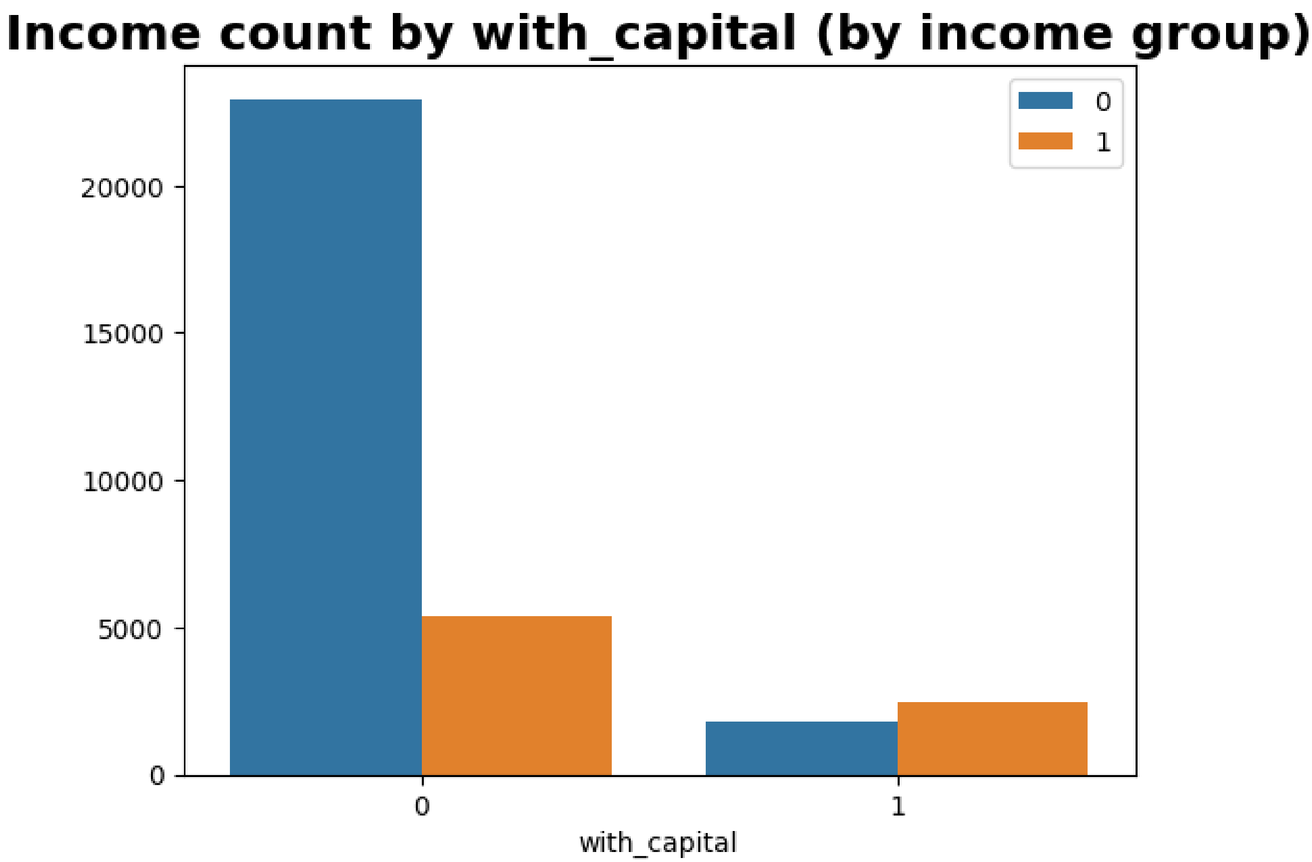

Figure 10: relation between capital and income

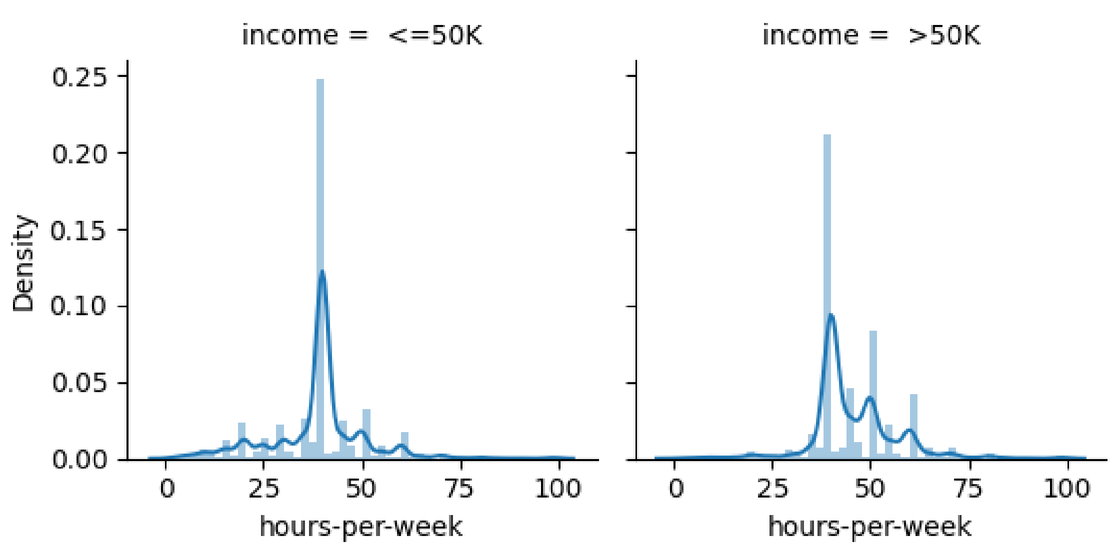

Figure 11: relation between hours-per-week and income

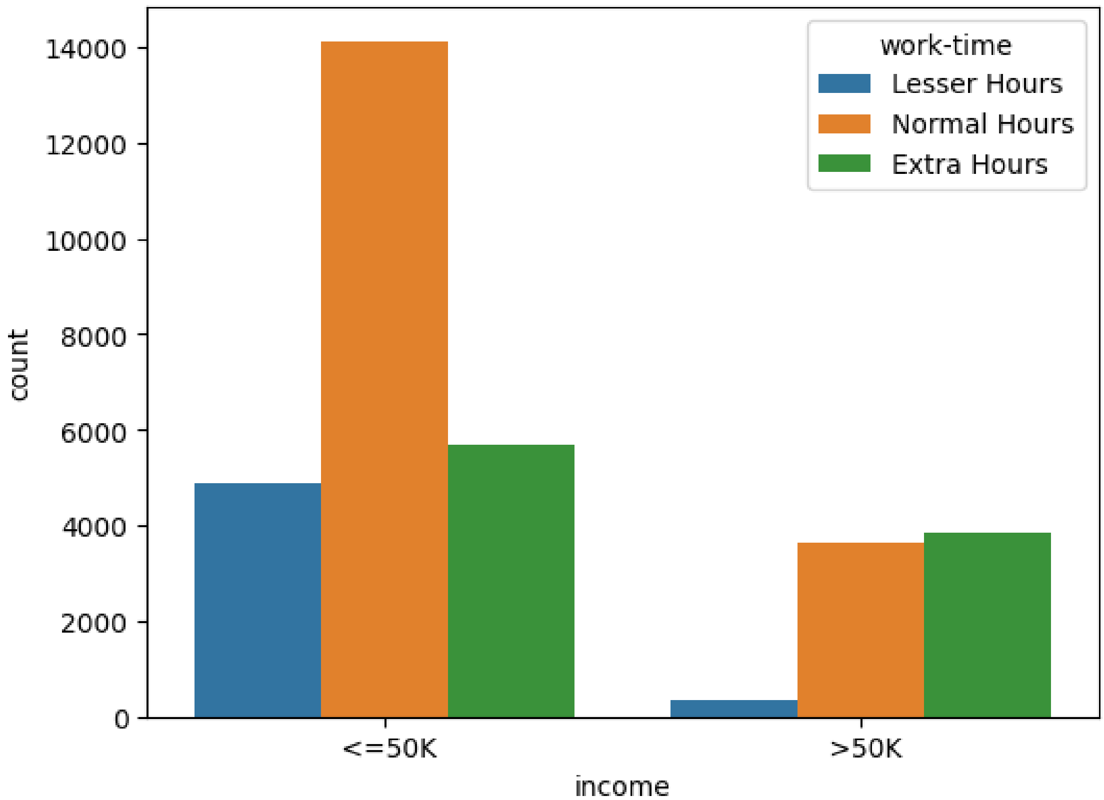

Figure 12: relation between work-time and income

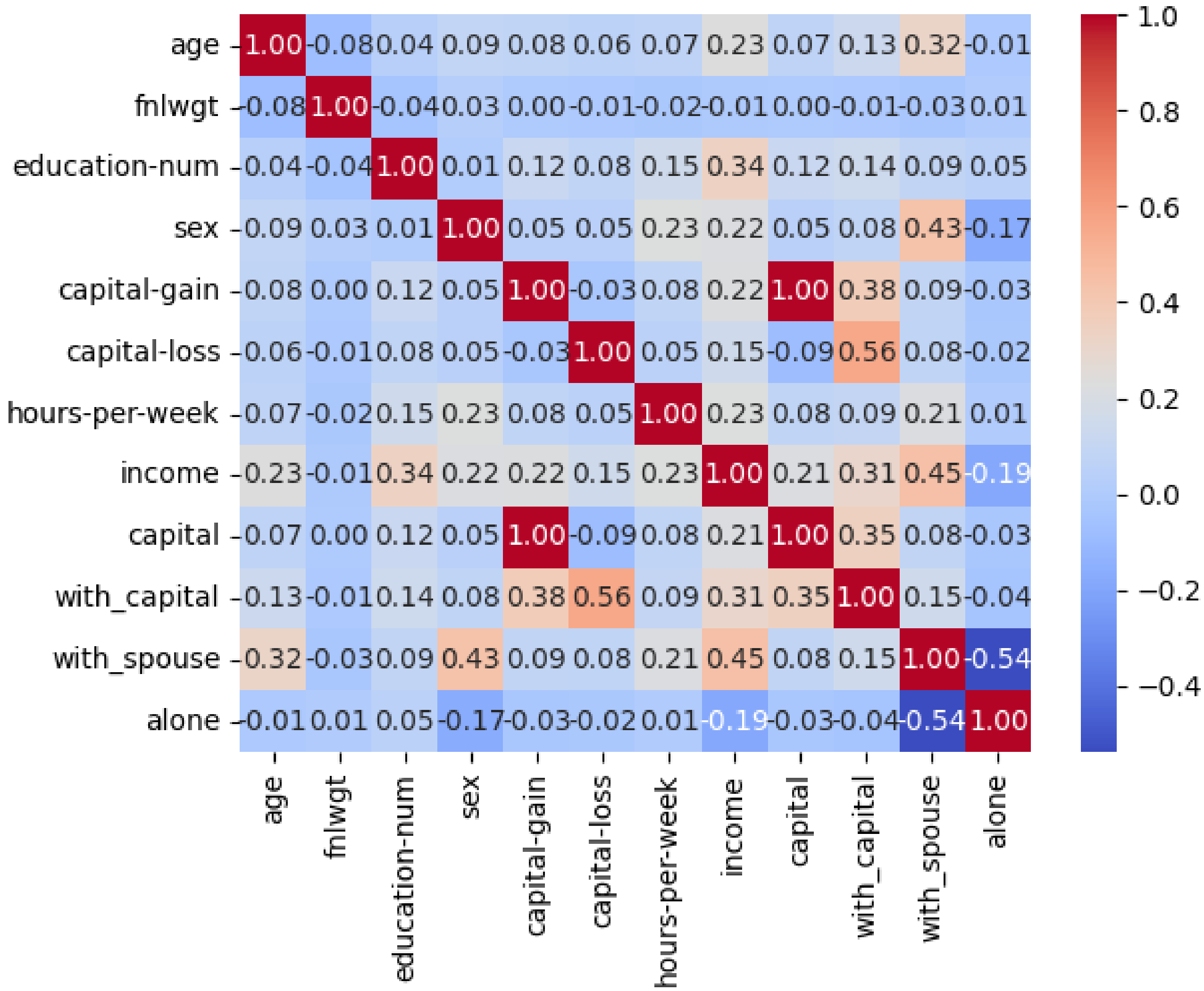

Figure 13: heat map with more features

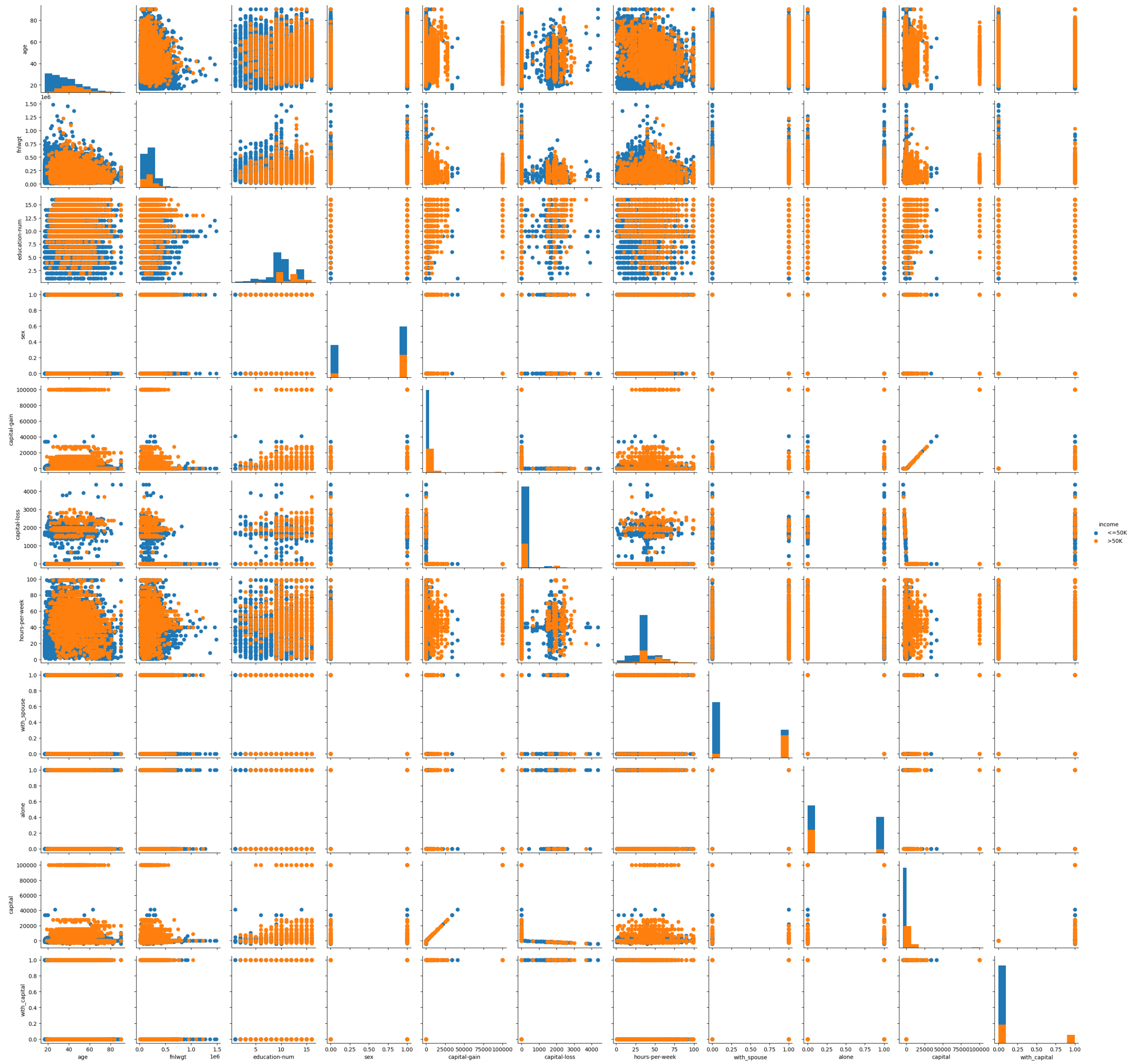

Figure 14: pairwise relationships

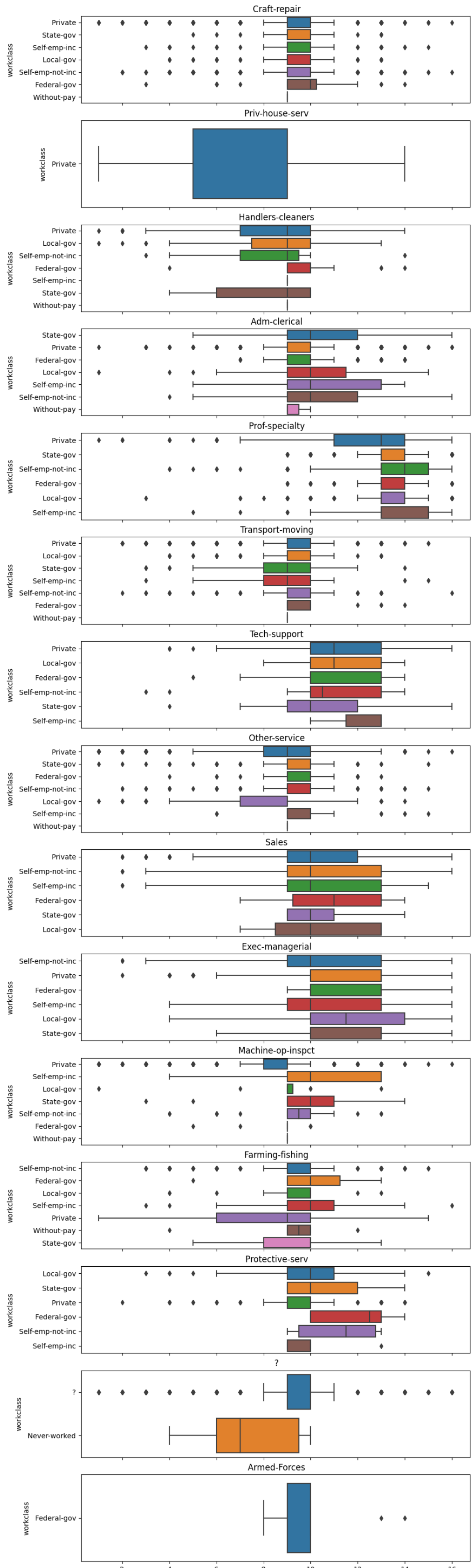

Figure 15: education-num vs occupation vs workclass

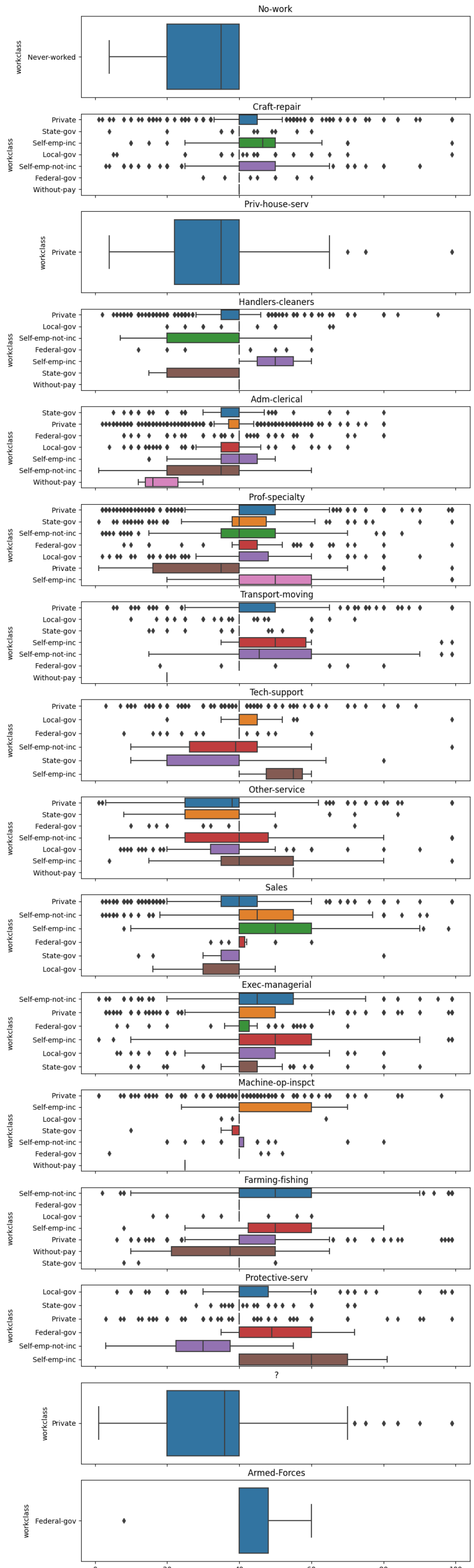

Figure 16: hours per week vs occupation vs workclass

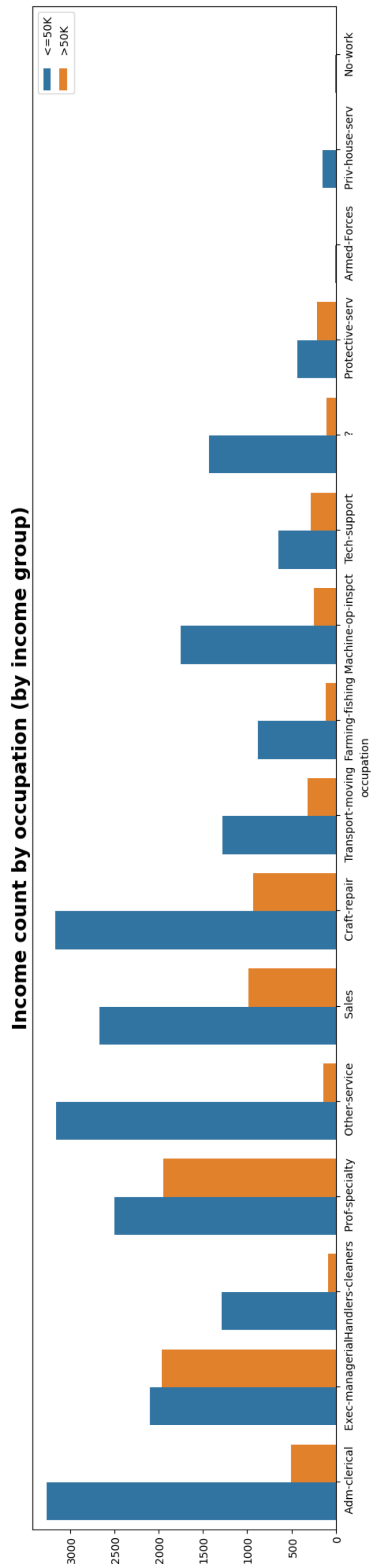

Figure 17: occupation vs income

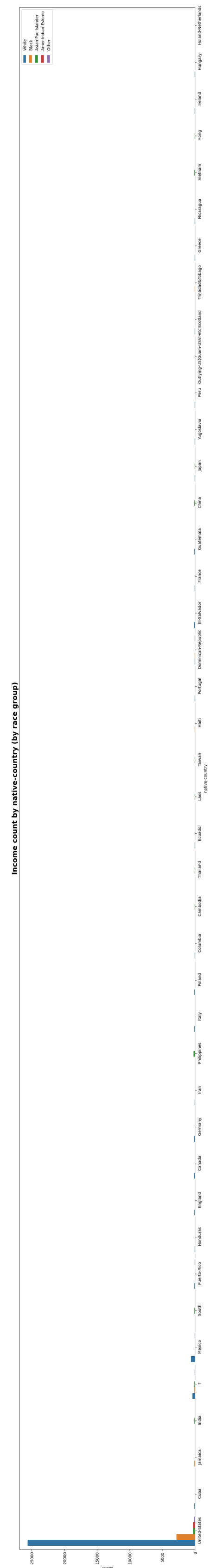

Figure 18: race vs country

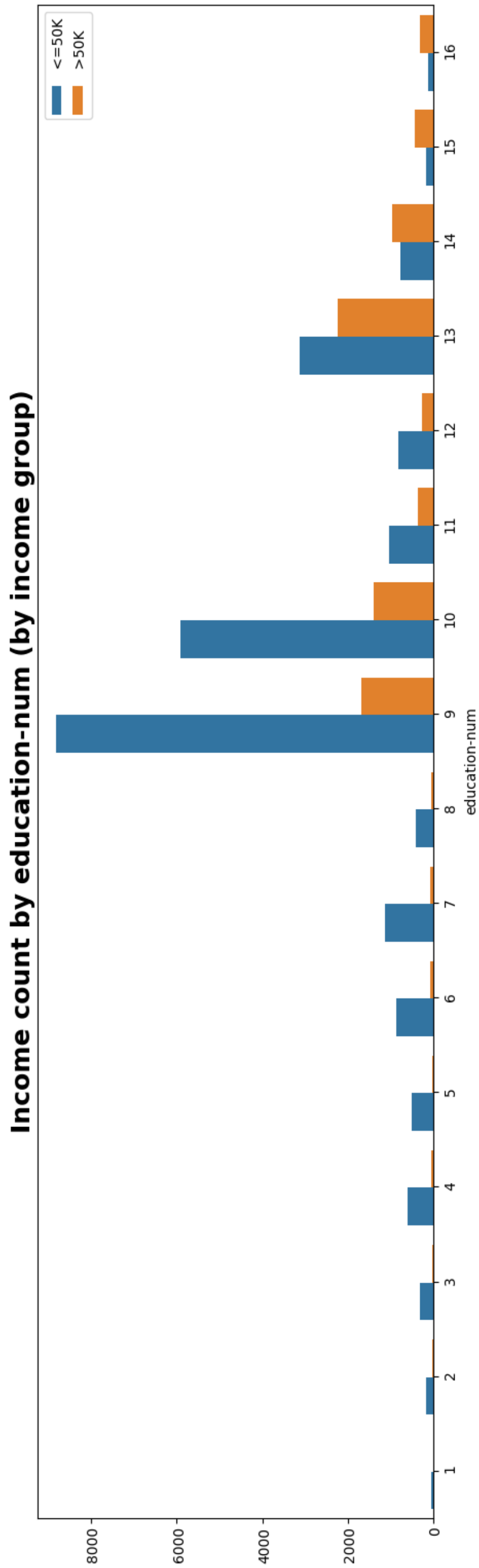

Figure 19: education-num vs income

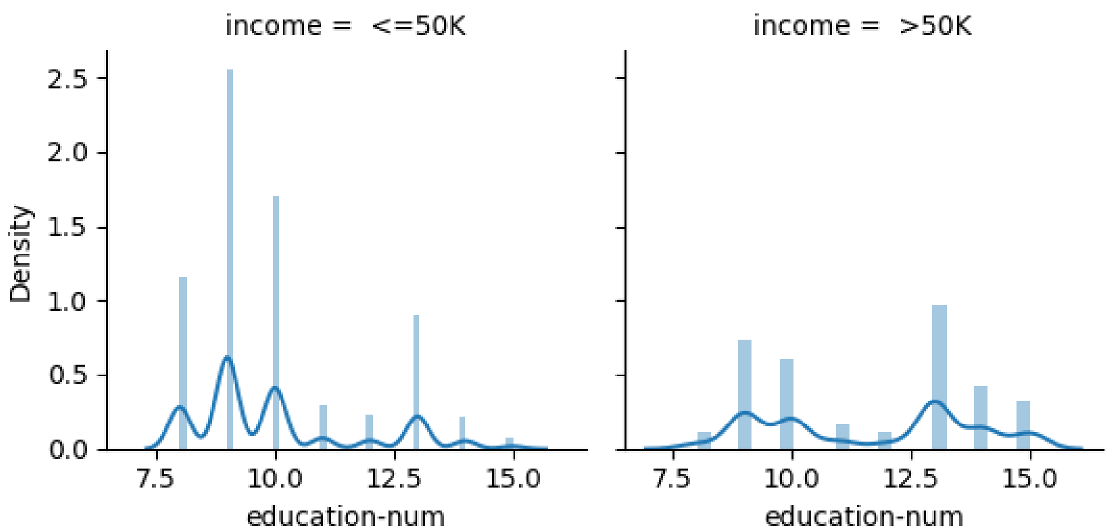

Figure 20: education-num vs income

```
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education-num   32561 non-null  int64
 4   marital-status  32561 non-null  object
 5   occupation      32561 non-null  object
 6   relationship    32561 non-null  object
 7   race            32561 non-null  object
 8   sex             32561 non-null  int64
 9   capital-gain    32561 non-null  int64
 10  capital-loss    32561 non-null  int64
 11  hours-per-week  32561 non-null  int64
 12  native-country  32561 non-null  object
 13  income          32561 non-null  int64
 14  with_spouse     32561 non-null  int64
 15  alone           32561 non-null  int64
 16  capital         32561 non-null  int64
 17  with_capital    32561 non-null  object
 18  work-time       32561 non-null  category
dtypes: category(1), int64(11), object(7)
```

Figure 21: features

	Model	Train_Score	Test_Score
0	Support Vector Machines	79.56	80.17
1	KNN	86.27	75.88
2	Logistic Regression	78.51	79.32
3	Random Forest	100.00	85.91
4	Decision Tree	100.00	81.20
5	GradientBoosting	87.79	87.30

Figure 22: model performance