

# Data Science Internship

BY MOHAMMAD ARSHADULLA NOOR  
AMINA MAHNOOR



# 02 Contents

**Introduction**

**Data Preprocessing**

**Data Visualisation**

**Model Building**

**Model Deployment**

**Conclusion**

## Problem Statement:

The aim is to visualize data and get insights from the dataset. To analyze how different categories play a role in income prediction.

## Introduction:

The adult income dataset that involves predicting personal income levels as above or below 50,000 per year based on personal details such as relationship and education level. This means that techniques for imbalanced classification can be used whilst model performance can still be reported using classification accuracy, as is used with balanced classification problems.

# Data Preprocessing

In this, you will discover how to develop and evaluate a model for the imbalanced adult income classification dataset.

We evaluate:

• How to load and explore the dataset and generate ideas for data preparation and model selection.

In this project, we use a standard imbalanced machine learning dataset referred to as the "Adult Income" dataset.

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspt	Husband	White	Male	0	0	40	United-States	>50K
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

# Data Preprocessing

The dataset provides 14 input variables that are a mixture of categorical, ordinal, and numerical data types. The complete list of variables is as follows:

```
In [7]: col_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital_status', 'occupation', 'relationship',  
                 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']  
df.columns = col_labels
```

The dataset contains missing values that are marked with a question mark character (?).

There are a total of 32,560 rows of data, and 15 with columns of data.

# Data Preprocessing

The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar indicates the notebook is titled 'Project' and is located at 'localhost:8888/notebooks/DST\_Internship/Project.ipynb#'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Help, Not Trusted, and Python 3 (ipykernel). The toolbar below the menu bar includes icons for cell operations like Run, Cell, Kernel, and Help.

In [4]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pickle
```

**Reading Dataset**

In [5]:

```
df = pd.read_csv('adult.data')
```

In [6]:

```
df
```

Out[6]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32555	27	Private	257302	Assoc-adm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspect	Husband	White	Male	0	0	40	United-States	>50K
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

32560 rows × 15 columns

# Data Preprocessing

The screenshot shows a Jupyter Notebook interface with multiple tabs at the top: 'ML API', 'DST\_Internship/', 'Project - Jupyter Notebook', and 'Project'. The main content area displays the following code and its output:

```
In [7]: col_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']
df.columns = col_labels

In [8]: df
```

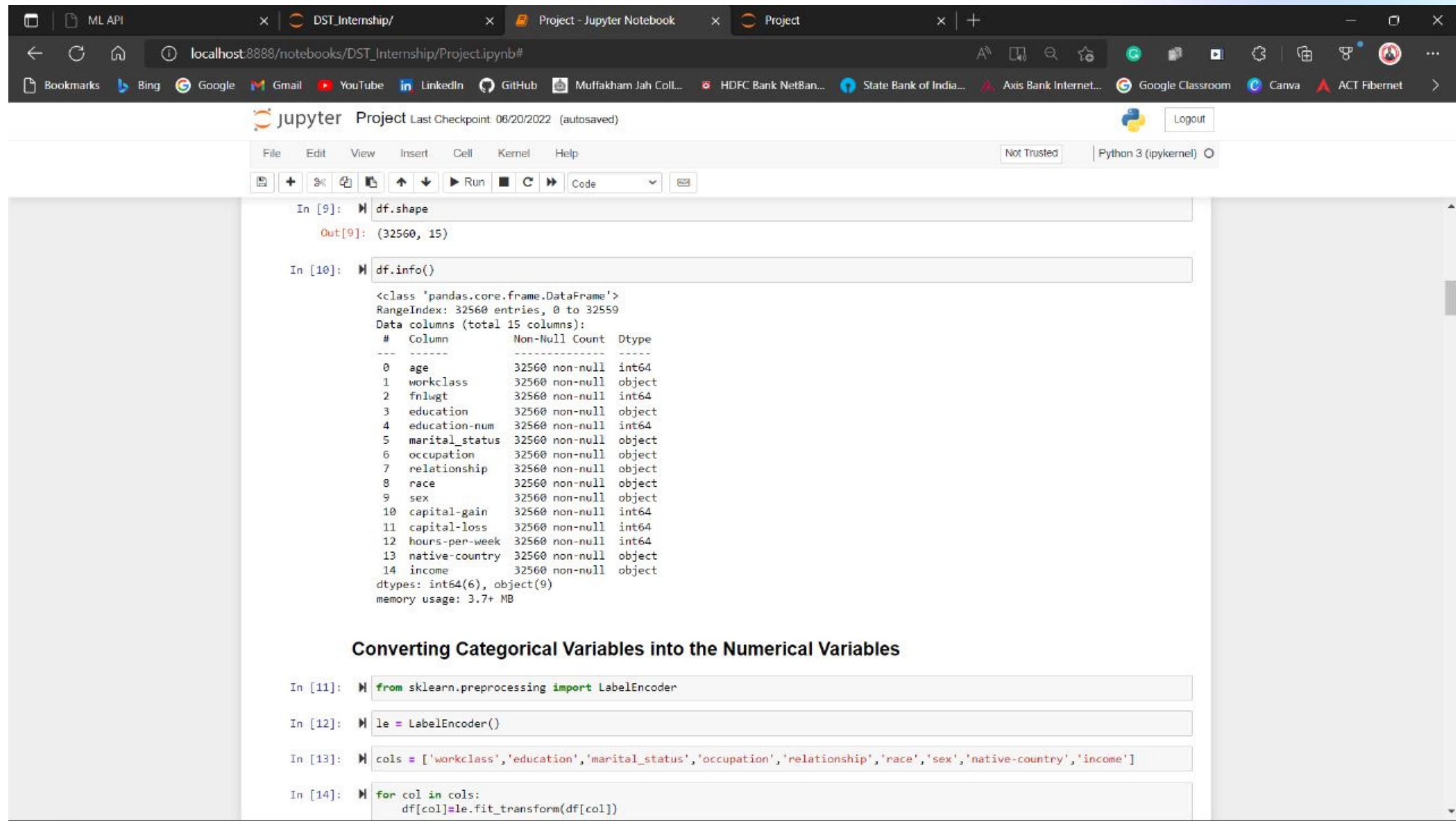
Out[8]:

	age	workclass	fnlwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-insct	Husband	White	Male	0	0	40	United-States	>50K
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

32560 rows × 15 columns

```
In [9]: df.shape
```

# Data Preprocessing



The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar indicates the notebook is titled "Project - Jupyter Notebook". The URL in the address bar is "localhost:8888/notebooks/DST\_Internship/Project.ipynb#". The notebook contains the following code:

```
In [9]: df.shape
Out[9]: (32560, 15)

In [10]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32560 entries, 0 to 32559
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   age         32560 non-null   int64  
 1   workclass    32560 non-null   object  
 2   fnlwgt       32560 non-null   int64  
 3   education    32560 non-null   object  
 4   education-num 32560 non-null   int64  
 5   marital_status 32560 non-null   object  
 6   occupation   32560 non-null   object  
 7   relationship  32560 non-null   object  
 8   race         32560 non-null   object  
 9   sex          32560 non-null   object  
 10  capital-gain 32560 non-null   int64  
 11  capital-loss 32560 non-null   int64  
 12  hours-per-week 32560 non-null   int64  
 13  native-country 32560 non-null   object  
 14  income        32560 non-null   object  
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

**Converting Categorical Variables into the Numerical Variables**

```
In [11]: from sklearn.preprocessing import LabelEncoder
In [12]: le = LabelEncoder()
In [13]: cols = ['workclass','education','marital_status','occupation','relationship','race','sex','native-country','income']
In [14]: for col in cols:
           df[col]=le.fit_transform(df[col])
```

# Data Visualization:

After data preprocessing, data visualization is done using the given dataset to check to analyze the income for different categories.

There are two class values ' $>50K$ ' and ' $\leq 50K$ ', meaning it is a binary classification task. The classes are imbalanced, with a skew toward the ' $\leq 50K$ ' class label.

' $>50K$ ': majority class, approximately 25%.

' $\leq 50K$ ': minority class, approximately 75%.

Different data visualizations are applied to the dataset like bargraph, heatmap, countplot to analyze the income for different categories.

# Data Visualization:

The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar has tabs for 'ML API', 'DST\_Internship/', 'Project - Jupyter Notebook', and 'Project'. The address bar shows 'localhost:8888/notebooks/DST\_Internship/Project.ipynb#'. Below the address bar is a toolbar with various icons for file operations, search, and navigation. The main content area contains two code cells.

**In [15]:** df

**Out[15]:**

	age	workclass	tfnwgt	education	education-num	marital_status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	50	6	83311	9	13	2	4	0	4	1	0	0	13	39	0
1	38	4	215646	11	9	0	6	1	4	1	0	0	40	39	0
2	53	4	234721	1	7	2	6	0	2	1	0	0	40	39	0
3	28	4	338409	9	13	2	10	5	2	0	0	0	40	5	0
4	37	4	284582	12	14	2	4	5	4	0	0	0	40	39	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32555	27	4	257302	7	12	2	13	5	4	0	0	0	38	39	0
32556	40	4	154374	11	9	2	7	0	4	1	0	0	40	39	1
32557	58	4	151910	11	9	6	1	4	4	0	0	0	40	39	0
32558	22	4	201490	11	9	4	1	3	4	1	0	0	20	39	0
32559	52	5	287927	11	9	2	4	5	4	0	15024	0	40	39	1

32560 rows × 15 columns

**Data Visualization**

**In [16]:** sns.factorplot(x="workclass", y="income", data=df, kind="bar", size = 6, palette = "muted")

C:\Users\HOTSHOT\AppData\Local\Programs\Python\Python39\lib\site-packages\seaborn\categorical.py:3717: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`. warnings.warn(msg) C:\Users\HOTSHOT\AppData\Local\Programs\Python\Python39\lib\site-packages\seaborn\categorical.py:3723: UserWarning: The `size` parameter has been renamed to `height`; please update your code. warnings.warn(msg, UserWarning)

**Out[16]:** <seaborn.axisgrid.FacetGrid at 0x1e7aa32f2e0>

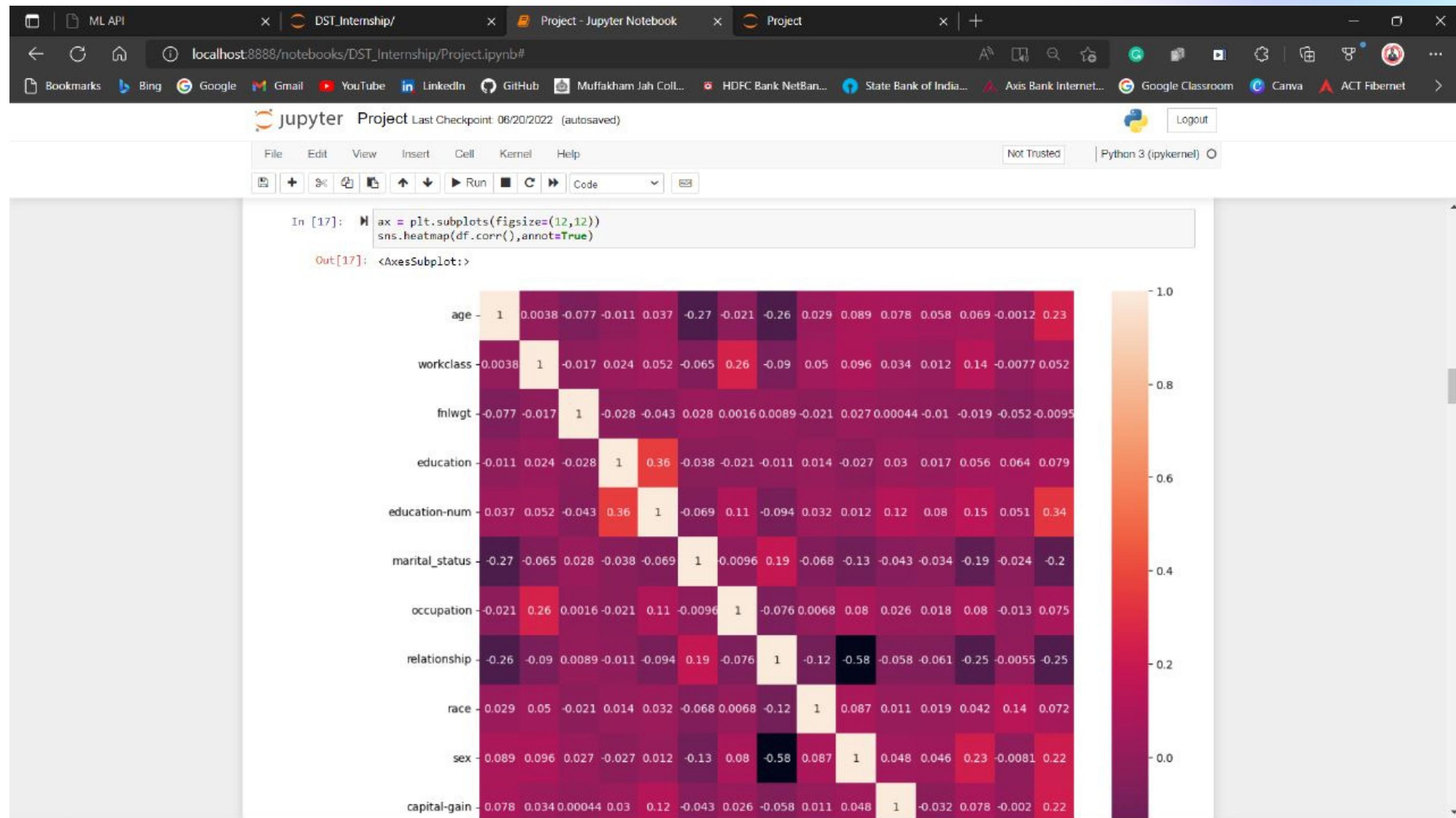
# Data Visualization:

The screenshot shows a Jupyter Notebook interface with the following details:

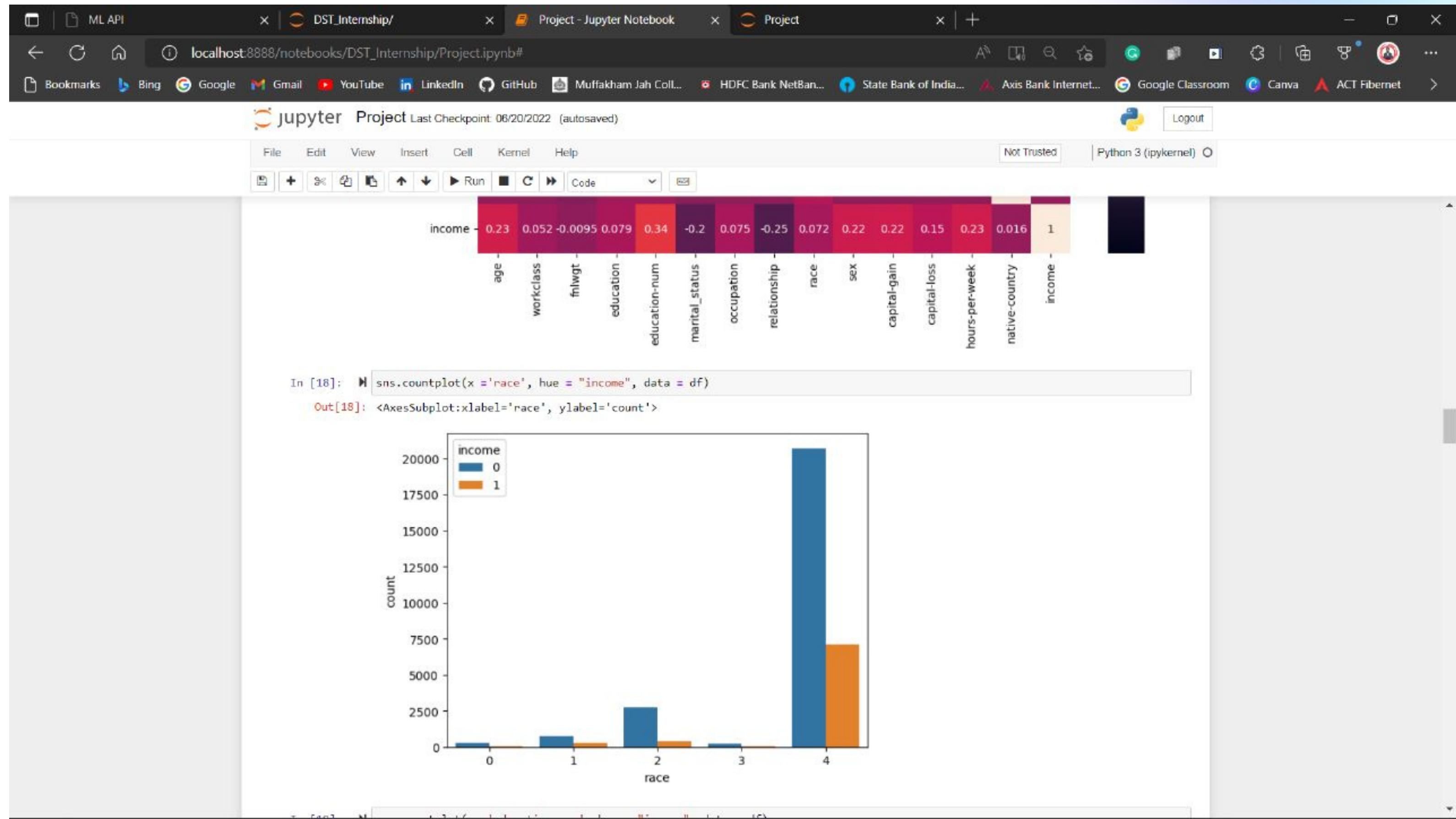
- Header:** The top bar includes tabs for "ML API", "DST\_Internship/", "Project - Jupyter Notebook", "Project", and a "+" button. Below the tabs is a toolbar with icons for back, forward, search, and file operations.
- Title Bar:** The title bar displays the URL "localhost:8888/notebooks/DST\_Internship/Project.ipynb#".
- User Information:** A "jupyter" icon, the word "Project", and "Last Checkpoint: 06/20/2022 (autosaved)" are shown.
- Toolbar:** A standard Jupyter toolbar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help" menus, along with "Not Trusted" and "Python 3 (ipykernel)" status indicators.
- Code Editor:** A code editor with a toolbar above it containing icons for file operations, run cell, and code cell.
- Output Cells:** Two output cells are visible:
  - Out[16]:** A bar chart titled "FacetGrid" showing the relationship between "workclass" (x-axis, categories 0-8) and "income" (y-axis, ranging from 0.0 to 0.6). The bars are colored blue, orange, green, purple, brown, pink, and grey.
  - In [17]:** A code cell containing: 

```
ax = plt.subplots(figsize=(12,12))  
sns.heatmap(df.corr(), annot=True)
```
  - Out[17]:** A heatmap titled "AxesSubplot" showing the correlation matrix of the dataset. The x and y axes are labeled with column names: "age", "workclass", "fnlwgt", "education", "education-num", "marital-status", "occupation", "relationship", "race", "sex", "capital-gain", "capital-loss", "hours-per-week", and "income". The heatmap uses a color scale from -0.8 (dark red) to 1.0 (yellow).

# Data Visualization:



# Data Visualization:



# Model Building:

Model Building is done using different types of classification models. Using predefined train and test sets, reported good testing classification accuracy of about 87.30 percent by using XGB Classifier.

This might provide a target to aim for when working on this dataset.

# Model Building:

The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar indicates the notebook is titled "Project - Jupyter Notebook". The browser tabs show other open notebooks like "ML API", "DST\_Internship/", and "Project". The address bar shows the URL "localhost:8888/notebooks/DST\_Internship/Project.ipynb#". The notebook content is organized into two main sections: "Splitting the Dataset in Train and Test" and "Using Different Classification Models".

**Splitting the Dataset in Train and Test**

```
In [22]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.25,random_state=3)

In [23]: from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.neural_network import MLPClassifier
        from sklearn.naive_bayes import GaussianNB
        from sklearn.tree import DecisionTreeClassifier
        import xgboost as xgb
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import model_selection
        from sklearn.metrics import accuracy_score
```

**Using Different Classification Models**

```
In [24]: models = []
        names = ["Random Forest Tranning Accuracy","Neural Network Tranning Accuracy","GaussianNB Tranning Accuracy",
                 "DecisionTreeClassifier Tranning Accuracy","XGB Tranning Accuracy","KNN Tranning Accuracy"]
        names2 = ["Random Forest Testing Accuracy","Neural Network Testing Accuracy","GaussianNB Testing Accuracy",
                  "DecisionTreeClassifier Testing Accuracy","XGB Testing Accuracy","KNN Testing Accuracy"]
        models.append((RandomForestClassifier(n_estimators=100)))
        models.append((MLPClassifier(max_iter=500)))
        models.append((GaussianNB()))
        models.append((DecisionTreeClassifier()))
        models.append((xgb.XGBClassifier()))
        models.append((KNeighborsClassifier()))

In [25]: kfold = model_selection.KFold(n_splits=5)

In [26]: for i in range(0,len(models)):
        cv_result = model_selection.cross_val_score(models[i],x_train,y_train,cv=kfold,scoring='accuracy')
        score=models[i].fit(x_train,y_train)
        training_acc = models[i].score(x_train,y_train)
        prediction = models[i].predict(x_test)
        acc_score_test = accuracy_score(y_test,prediction)
        print ('*' * 100)
```

# Model Building:

The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar indicates the notebook is titled 'Project - Jupyter Notebook'. The URL in the address bar is 'localhost:8888/notebooks/DST\_Internship/Project.ipynb#'. The notebook contains several cells of Python code.

**In [26]:**

```
for i in range(0,len(models)):
    cv_result = model_selection.cross_val_score(models[i],x_train,y_train,cv=kfold,scoring='accuracy')
    score=models[i].fit(x_train,y_train)
    training_acc = models[i].score(x_train,y_train)
    prediction = models[i].predict(x_test)
    acc_score_test = accuracy_score(y_test,prediction)
    print ('**'*40)
    print ('{0}: {1}'.format(names[i],training_acc))
    print ('{0}: {1}'.format(names2[i],acc_score_test))

*****
Random Forest Tranining Accuracy: 0.9998771498771499
Random Forest Testing Accuracy: 0.8544226044226844
*****
Neural Network Tranining Accuracy: 0.7934479934479934
Neural Network Testing Accuracy: 0.7907862407862408
*****
GaussianNB Tranining Accuracy: 0.7968877968877969
GaussianNB Testing Accuracy: 0.7925061425061425
*****
DecisionTreeClassifier Tranining Accuracy: 0.9999590499590499
DecisionTreeClassifier Testing Accuracy: 0.8101965601965602
*****
XGB Tranining Accuracy: 0.9115069615069615
XGB Testing Accuracy: 0.8707616707616708
*****
KNN Tranining Accuracy: 0.8337018837018837
KNN Testing Accuracy: 0.7787469287469287
```

**In [27]:**

```
from sklearn.model_selection import GridSearchCV
```

**Using GridSearchCV It is a Technique to Search Through the Best Parameter Values**

**In [28]:**

```
learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
param_grid = dict(learning_rate=learning_rate)
model = xgb.XGBClassifier()
grid_search = GridSearchCV(model, param_grid, scoring="neg_log_loss", n_jobs=-1, cv=10)
grid_search.fit(x_train, y_train)
```

**Out[28]:** GridSearchCV(cv=10,

# Model Building:

The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar has tabs for 'ML API', 'DST\_Internship/', 'Project - Jupyter Notebook', and 'Project'. The main content area displays Python code for building an XGBClassifier model using GridSearchCV. The code defines the estimator, parameters, and scoring metric. It also shows the assignment of the best estimator from the grid search to a variable 'best\_model'. Below this, a section titled 'Testing of XGB Classification Model (as it gave best result above)' is shown, with code to fit the model to training data.

```
Out[28]: GridSearchCV(cv=10,
                      estimator=XGBClassifier(base_score=None, booster=None,
                                             callbacks=None, colsample_bylevel=None,
                                             colsample_bynode=None,
                                             colsample_bytree=None,
                                             early_stopping_rounds=None,
                                             enable_categorical=False, eval_metric=None,
                                             gamma=None, gpu_id=None, grow_policy=None,
                                             importance_type=None,
                                             interaction_constraints=None,
                                             learning_rate=None, max_bin=None,
                                             max_cat_to_onehot=None,
                                             max_delta_step=None, max_depth=None,
                                             max_leaves=None, min_child_weight=None,
                                             missing=nan, monotone_constraints=None,
                                             n_estimators=100, n_jobs=None,
                                             num_parallel_tree=None, predictor=None,
                                             random_state=None, reg_alpha=None,
                                             reg_lambda=None, ...),
                      n_jobs=-1,
                      param_grid={'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]},
                      scoring='neg_log_loss')

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [29]: top_param=grid_search.best_estimator_

In [30]: best_model=xgb.XGBClassifier(learning_rate=top_param.learning_rate, booster=top_param.booster,
                                     gamma=top_param.gamma, n_estimators=top_param.n_estimators)

Testing of XGB Classification Model (as it gave best result above)

In [31]: best_model.fit(x_train, y_train)

Out[31]: XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
                      colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
```

# Conclusion:

By looking above values we can say that education-num, marital status, relationship and capital gain are the features that are important for income prediction.

# Model Deployment:

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** Project.
- Toolbars:** Standard window controls (minimize, maximize, close).
- Left Sidebar (EXPLORER):**
  - OPEN EDITORS: app.py, index.html, result.html (highlighted).
  - PROJECT:
    - \_\_pycache\_\_
    - static
      - style.css
      - templates
        - index.html
        - result.html (highlighted)
    - venv
      - adult.data
      - app.py
      - Project.ipynb
      - Project.pkl
- Central Area:** Code editor showing the content of result.html.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>{{ prediction }}</h1>
  </body>
</html>
```
- Bottom Status Bar:** Ln 1, Col 1, Spaces: 4, UTF-8, CRLF, HTML, Go Live, Stylelint+, Colorize: 0 variables, Colorize, Prettier+, Prettier, Live Share.

# Model Deployment:

The screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Includes icons for back, forward, search, and project navigation.
- Project Explorer:** Shows the project structure:
  - PROJECT**:
    - \_\_pycache\_\_
    - static
      - style.css
    - templates
      - index.html
      - result.html
    - venv
    - adult.data
    - app.py
    - Project.ipynb
    - Project.pkl
- Open Editors:** app.py (selected), index.html, result.html.
- Code Editor:** Displays the content of app.py, which is a Flask application for predicting income based on the adult dataset. The code includes routes for home, ValuePredictor, result, and a POST route for result. It uses pickle to load a trained model from Project.pkl and performs predictions on input data.

# Model Deployment:

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** Project.
- Toolbars:** Standard icons for file operations.
- Left Sidebar (EXPLORER):**
  - OPEN EDITORS: app.py, index.html (selected), result.html
  - PROJECT:
    - \_pycache\_
    - static
      - style.css
    - templates
      - index.html (selected)
      - result.html
    - venv
    - adult.data
    - app.py
    - Project.ipynb
    - Project.pkl
- Central Area:** Content of index.html file.

```
1  <!DOCTYPE html>
2  <html>
3  <!--From https://codepen.io/frytyler/pen/EGdtg-->
4  <head>
5    <meta charset="UTF-8">
6    <title>ML API</title>
7    <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
8    <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
9    <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
10   <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
11   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}>
12 </head>
13
14 <body>
15   <h3>Income Prediction Form</h3>
16
17   <div>
18     <form action="/result" method="POST">
19       <label for="education-num">Education Number</label>
20       <input type="text" id="education-num" name="education-num">
21       <br>
22       <label for="marital_stat">Marital Status</label>
23       <select id="marital_stat" name="marital_stat">
24         <option value="0">divorced</option>
25         <option value="1">married</option>
26         <option value="2">not married</option>
27       </select>
28       <br>
29       <label for="occup">Occupation</label>
30       <select id="occup" name="occup">
31         <option value="0">Adm-clerical</option>
32         <option value="1">Armed-Forces</option>
33         <option value="2">Craft-repair</option>
34         <option value="3">Exec-managerial</option>
35         <option value="4">Farming-fishing</option>
36         <option value="5">Handlers-cleaners</option>
37         <option value="6">Machine-op-inspect</option>
38         <option value="7">Other-service</option>
39         <option value="8">Sales-service</option>
40       </select>
41     </form>
42   </div>
43
44 </body>
45 </html>
```
- Bottom Status Bar:** Ln 26, Col 39, Spaces: 2, UTF-8, CRLF, HTML, Go Live, Stylelint+, Colorize: 0 variables, Colorize, Prettier+, Prettier, Live Share.

# Model Deployment:

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Search Bar:** Q Project.
- Toolbar:** Standard window controls.
- Left Sidebar (EXPLORER):**
  - OPEN EDITORS: app.py, index.html (temp...), result.html (temp...).
  - PROJECT:
    - \_pycache\_
    - static
      - style.css
    - templates
      - index.html (selected)
      - result.html
    - venv
    - adult.data
    - app.py
    - Project.ipynb
    - Project.pkl
- Central Area:** Code editor showing index.html content.

```
templates > index.html > html > body > div > form > select#martial_stat > option
34     <option value="2">Craft-repair</option>
35     <option value="3">Exec-managerial</option>
36     <option value="4">Farming-fishing</option>
37     <option value="5">Handlers-cleaners</option>
38     <option value="6">Machine-op-inspect</option>
39     <option value="7">Other-service</option>
40     <option value="8">Priv-house-serv</option>
41     <option value="9">Prof-specialty</option>
42     <option value="10">Protective-serv</option>
43     <option value="11">Sales</option>
44     <option value="12">Tech-support</option>
45     <option value="13">Transport-moving</option>
46 </select>
47 <br>
48 <label for="relation">Relationship</label>
49 <select id="relation" name="relation">
50     <option value="0">Husband</option>
51     <option value="1">Not-in-family</option>
52     <option value="2">Other-relative</option>
53     <option value="3">Own-child</option>
54     <option value="4">Unmarried</option>
55     <option value="5">Wife</option>
56 </select>
57 <br>
58 <label for="c_gain">Capital Gain </label>
59 <input type="text" id="c_gain" name="c_gain">btw:[0-99999]
60 <br>
61 <label for="c_loss">Capital Loss </label>
62 <input type="text" id="c_loss" name="c_loss">btw:[0-4356]
63 <br>
64 <input type="submit" value="Submit">
65 </form>
66 </div>
67 </body>
68 </html>
```
- Right Sidebar:** Includes a 'RECENT' section with project files and a 'PROBLEMS' tab.

# RESULT:

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Back, Forward, Project search bar, and window controls.
- Left Sidebar (EXPLORER):**
  - OPEN EDITORS: app.py, index.html, result.html (highlighted).
  - PROJECT:
    - \_pycache\_
    - static
      - style.css
    - templates
      - index.html
      - result.html (highlighted)
    - venv
  - Other files: adult.data, app.py, Project.ipynb, Project.pkl.
- Right Editor Area:** Displays the content of the selected file, result.html:

```
1 <!DOCTYPE html>
2 <html>
3   <body>
4     <h1>{{ prediction }}</h1>
5   </body>
6 </html>
```

# RESULT:

The screenshot shows a web browser window with two tabs open. The left tab displays an 'Income Prediction Form' with various input fields and dropdown menus. The right tab shows the resulting prediction.

**Income Prediction Form (Left Tab):**

- Education Number: [Input field]
- Marital Status: divorced
- Occupation: Adm-clerical
- Relationship: Husband
- Capital Gain: btw:[0-99999]
- Capital Loss: Capital Loss
- btw:[0-4356]
- Submit

**Result (Right Tab):**

Income less than 50K