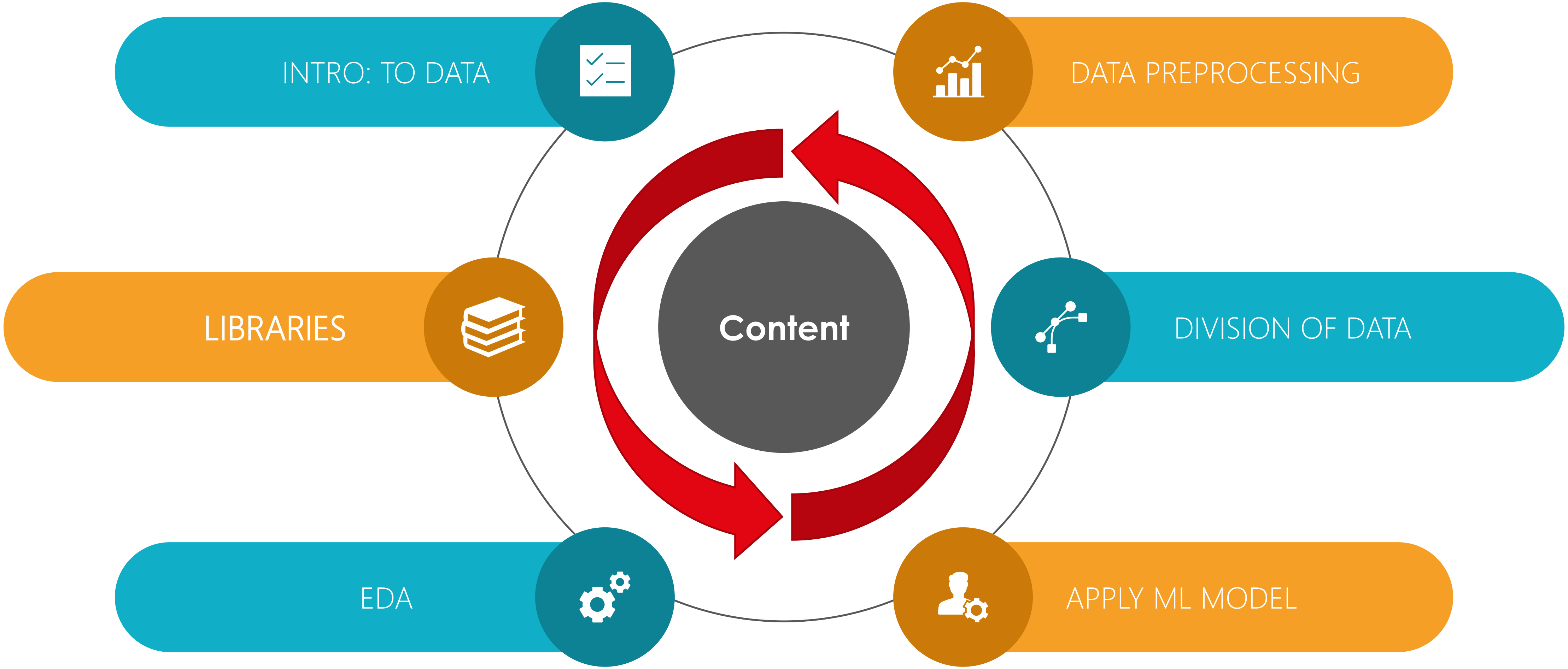
Classification of Census Income of Pakistan

By: Imran Ali

Contents:


INTRO: TO DATA
DATA PREPROCESSING
LIBRARIES
Content
DIVISION OF DATA
EDA
APPLY ML MODEL

INTRODUCTION TO DATA

DATA

Census income data of Pakistan Containing Following Columns & Values:

Listing of attributes:

- **salary:** >50K, <=50K
- **age:** continuous
- **workclass:** Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
- **fnlwgt:** continuous
- **education:** Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
- **education-num:** continuous
- **marital-status:** Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse

DATA

- **occupation:** Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
- **relationship:** Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
- **race:** White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
- **sex:** Female, Male
- **capital-gain:** continuous
- **capital-loss:** continuous
- **hours-per-week:** continuous
- **native-country:** Pakistan, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

LIBRARIES

python
Programming

Visual Studio Code

scikit
learn

pandas

seaborn

matplotlib

EDA & VISUALIZATION

PREPROCESSING

```
census = pd.read_csv('census-pk.csv')
census.head()
```

Python

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income	income_a
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	Pakistan	<=50K	
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	Pakistan	<=50K	
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	Pakistan	>50K	
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	Pakistan	>50K	
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	Pakistan	<=50K	

```
census.shape
```

4]

```
(43832, 16)
```

PREPROCESSING

```
census.info()
```

[120]

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 43832 entries, 0 to 48841
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             43832 non-null  int64 
 1   workclass       43832 non-null  object
 2   fnlwgt          43832 non-null  int64 
 3   education       43832 non-null  object
 4   education-num   43832 non-null  int64 
 5   marital-status  43832 non-null  object
 6   occupation      43832 non-null  object
 7   relationship    43832 non-null  object
 8   race            43832 non-null  object
 9   sex             43832 non-null  object
 10  capital-gain    43832 non-null  int64 
 11  capital-loss    43832 non-null  int64 
 12  hours-per-week  43832 non-null  int64 
 13  native-country  43832 non-null  object
 14  income          43832 non-null  object
 15  income_above_50K  43832 non-null  int64 
dtypes: int64(7), object(9)
memory usage: 5.7+ MB
```

PREPROCESSING

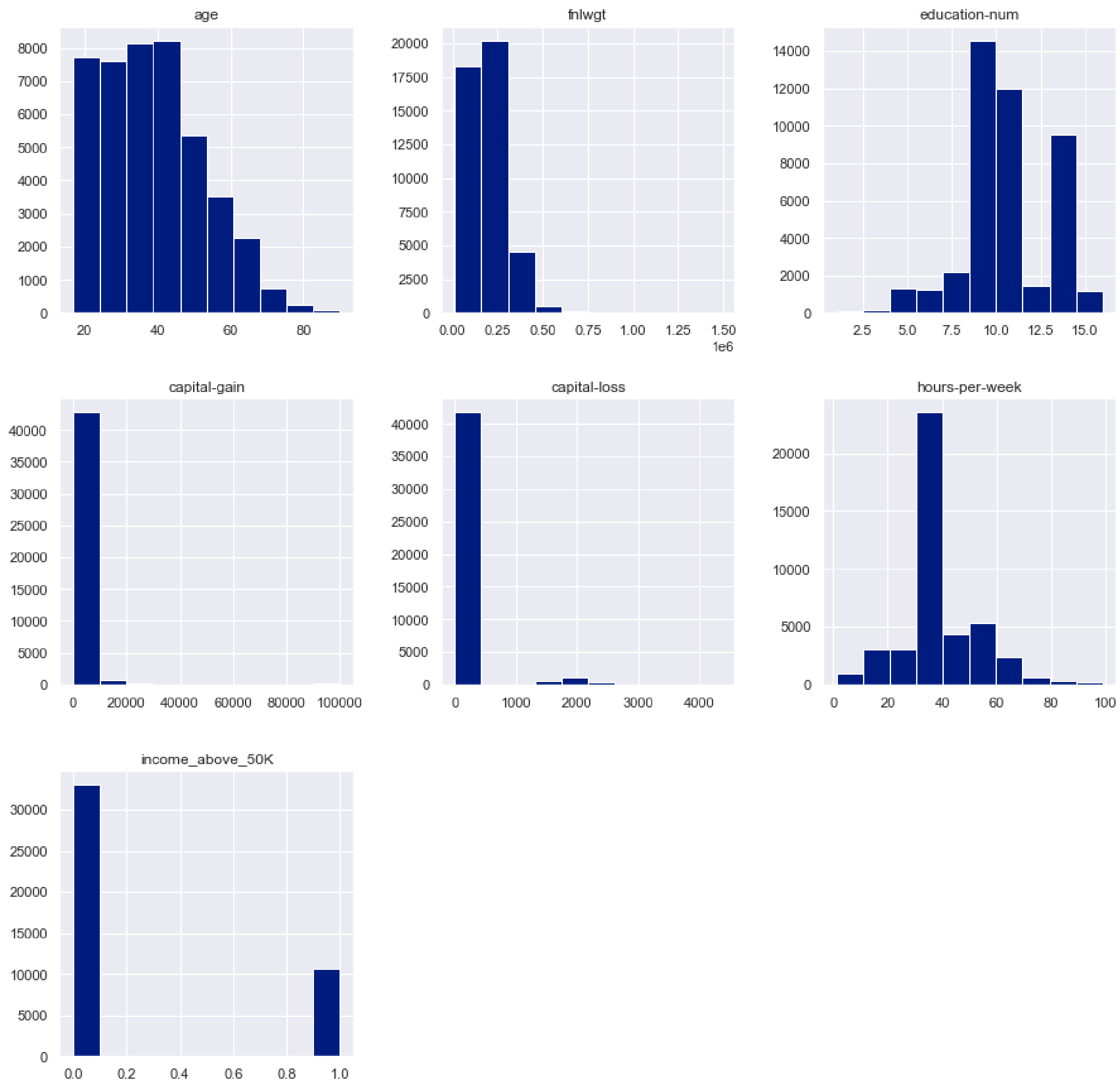
```
census.describe()
```

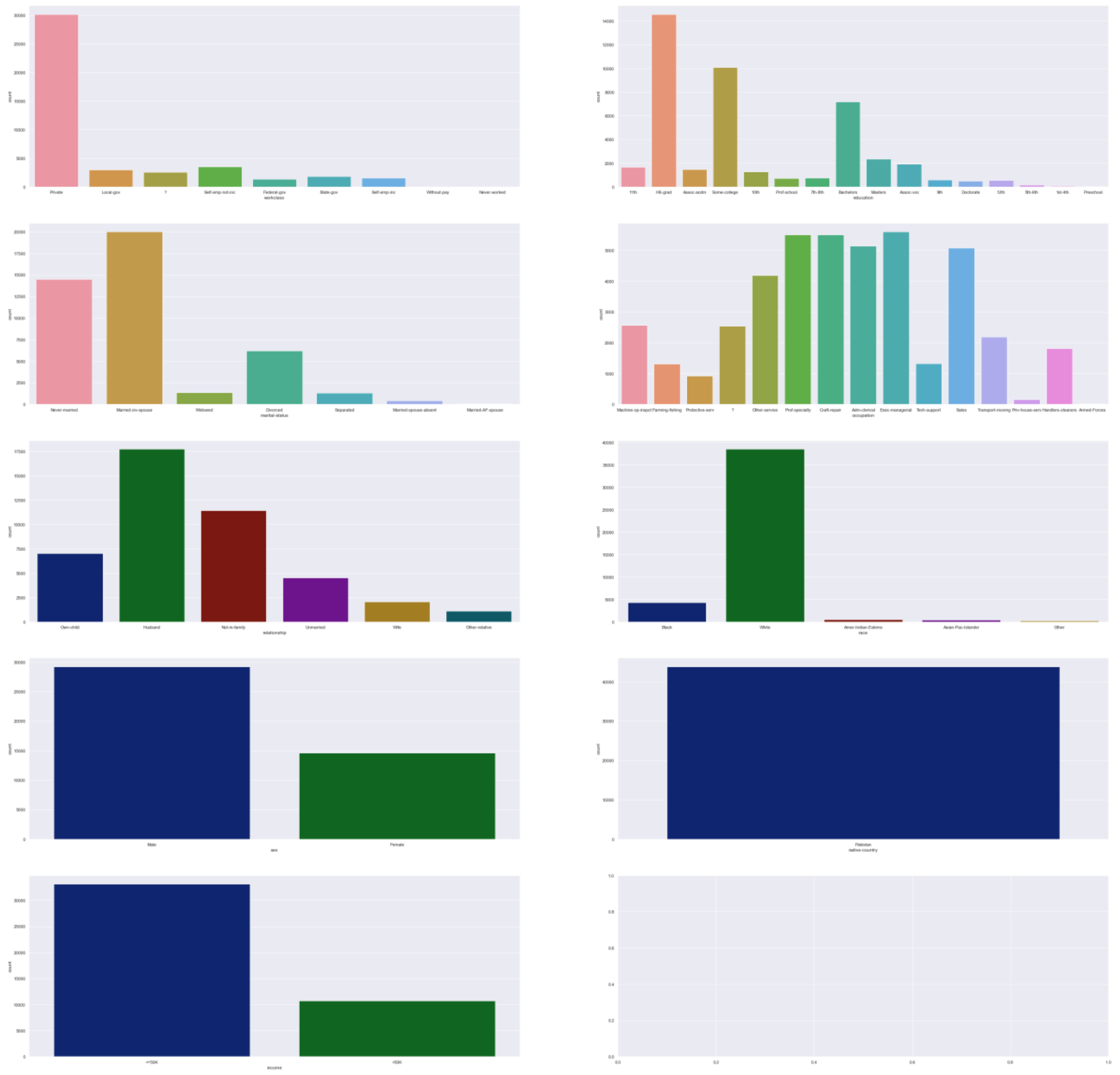
	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	income_above_50K
count	43832.000000	4.383200e+04	43832.000000	43832.000000	43832.000000	43832.000000	43832.000000
mean	38.698690	1.871419e+05	10.168667	1089.626529	88.789743	40.440774	0.243977
std	13.795294	1.051913e+05	2.393353	7455.791326	405.539243	12.471352	0.429484
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000	0.000000
25%	28.000000	1.156770e+05	9.000000	0.000000	0.000000	40.000000	0.000000
50%	37.000000	1.766720e+05	10.000000	0.000000	0.000000	40.000000	0.000000
75%	48.000000	2.344772e+05	12.000000	0.000000	0.000000	45.000000	0.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000	1.000000

```
#statistical sumaary -categorical features
census.describe(include=object)
```

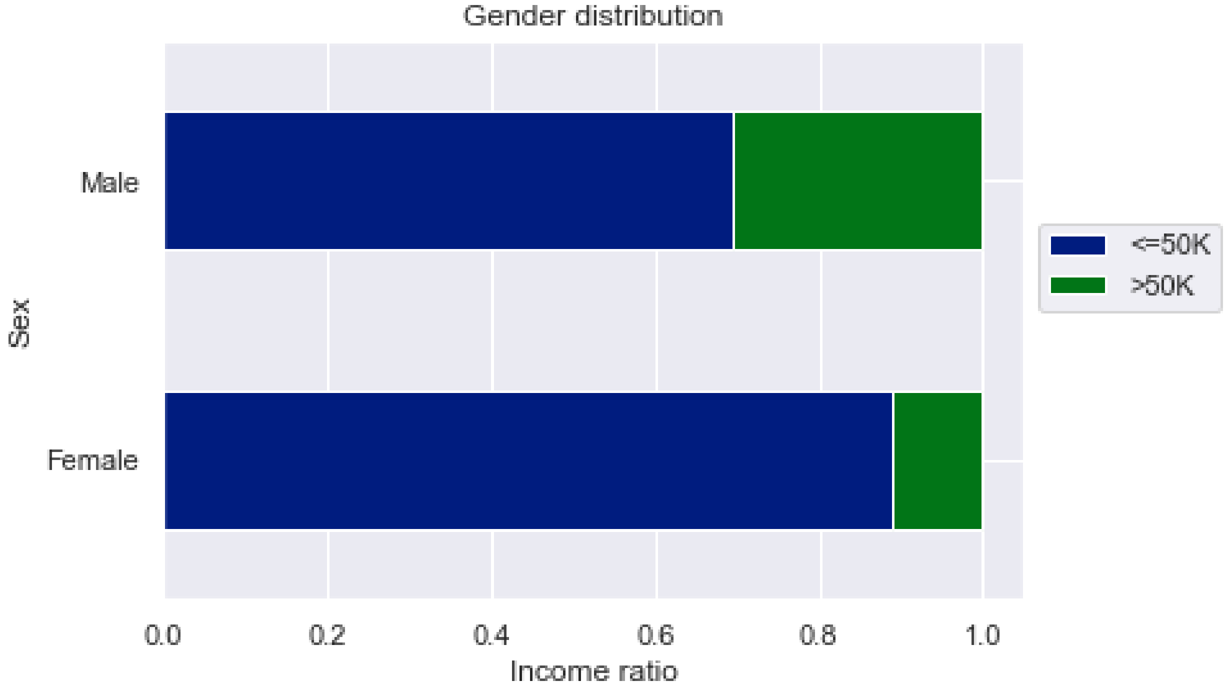
	workclass	education	marital-status	occupation	relationship	race	sex	native-country	income
count	43832	43832	43832	43832	43832	43832	43832	43832	43832
unique	9	16	7	15	6	5	2	1	2
top	Private	HS-grad	Married-civ-spouse	Exec-managerial	Husband	White	Male	Pakistan	<=50K
freq	30145	14570	20003	5606	17733	38493	29223	43832	33138

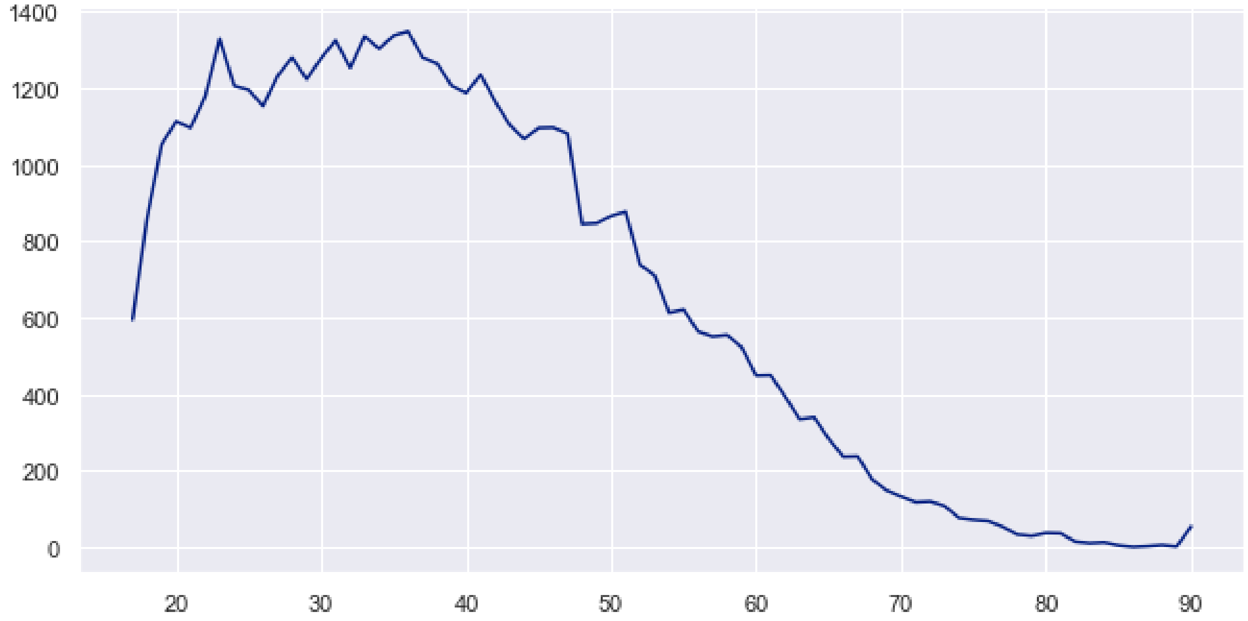
VISUALIZATION

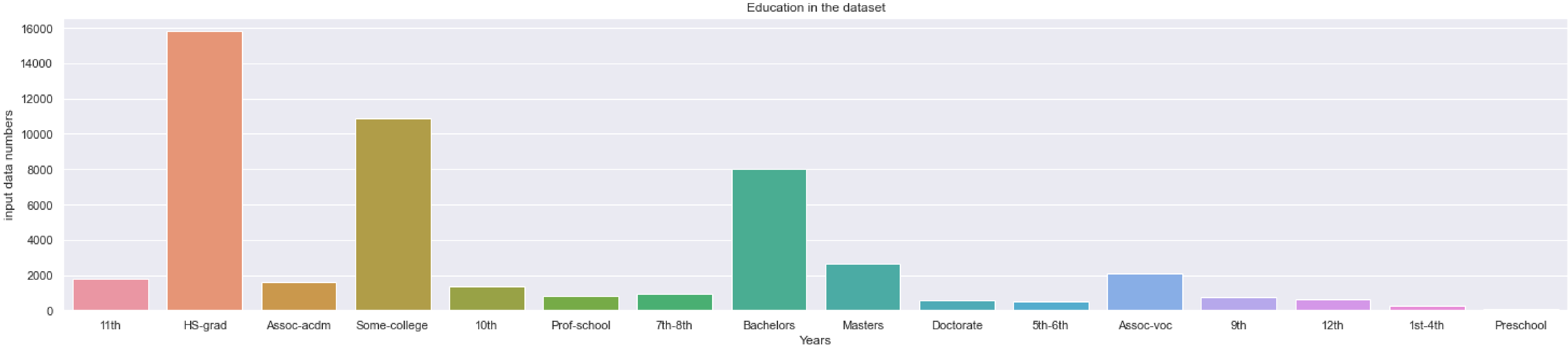




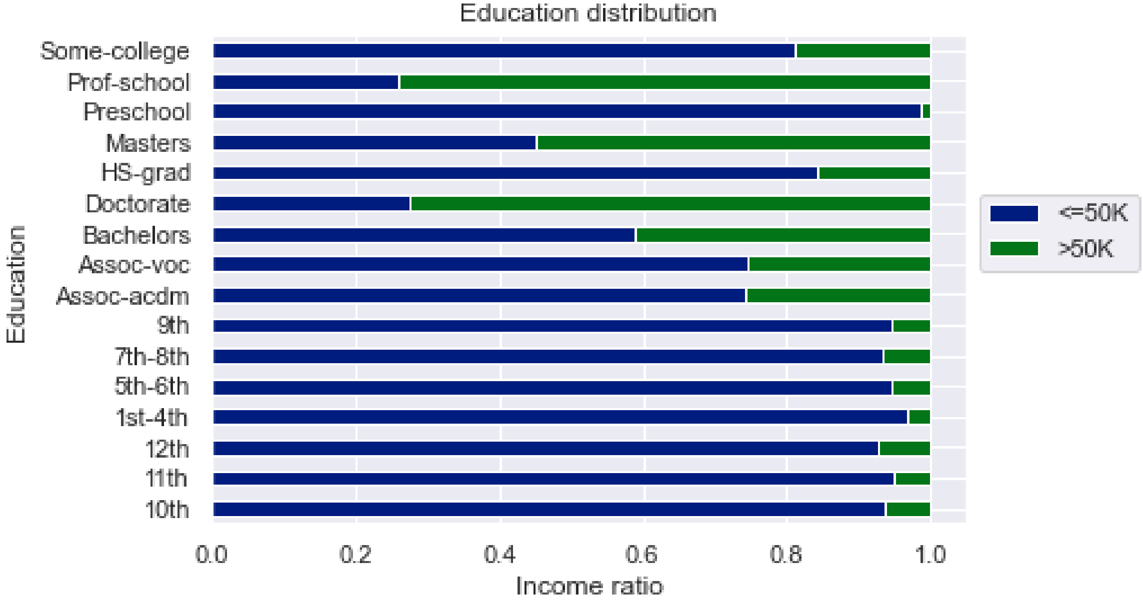
VISUALIZATION

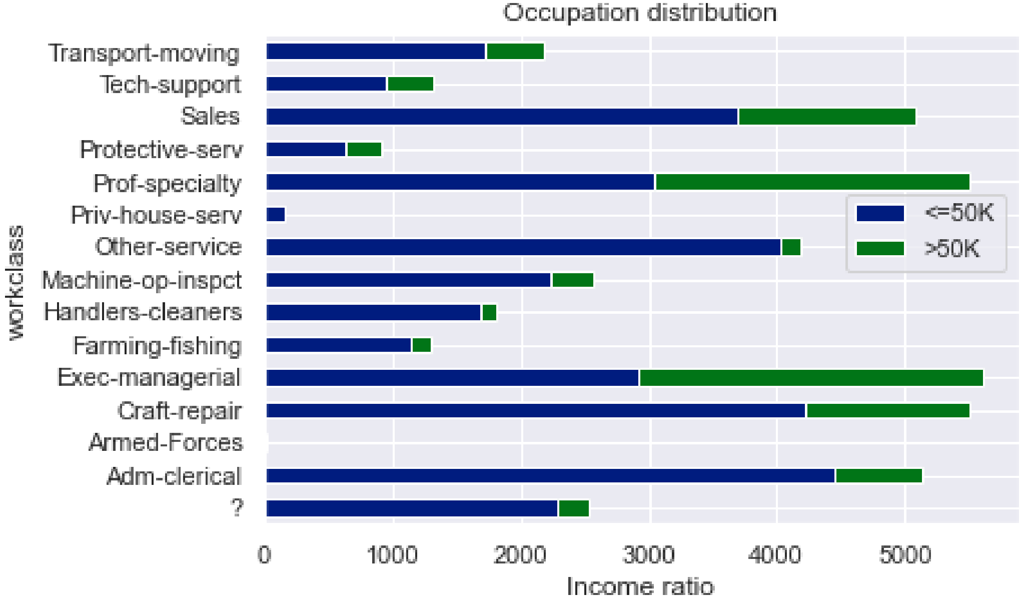


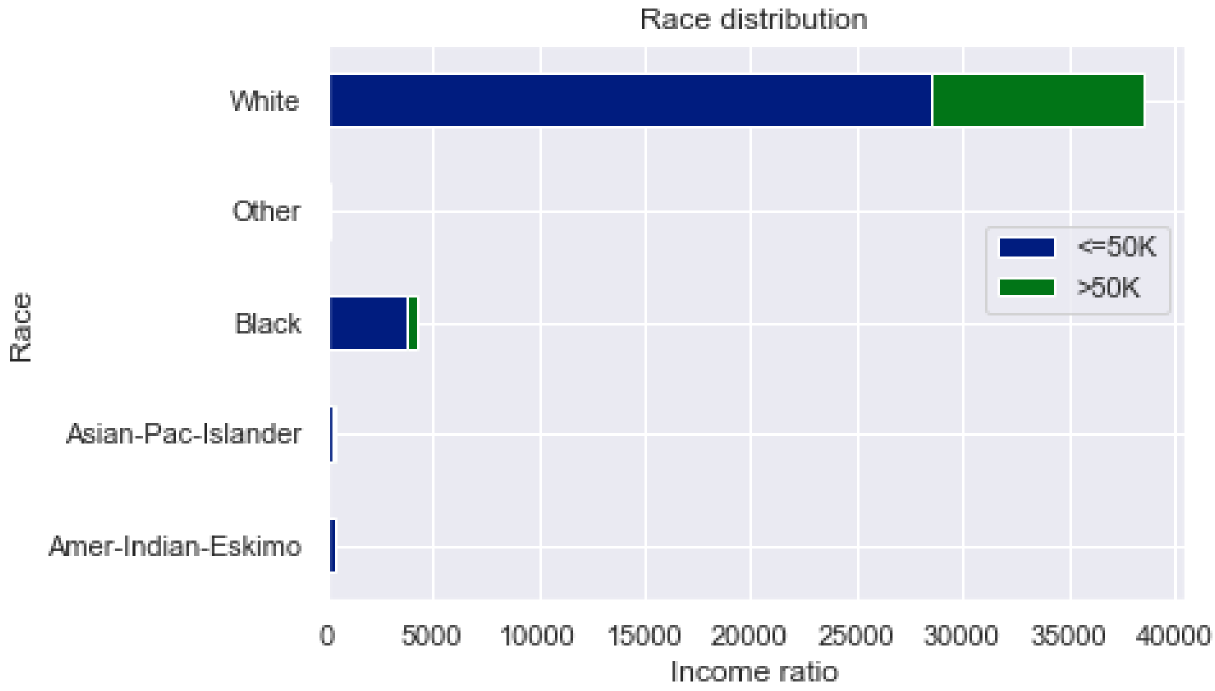




VISUALIZATION







DATA PREPROCESSING

PREPROCESSING

```python
census.drop(['income', 'native-country'], axis=1, inplace=True)
```

[138]

Python

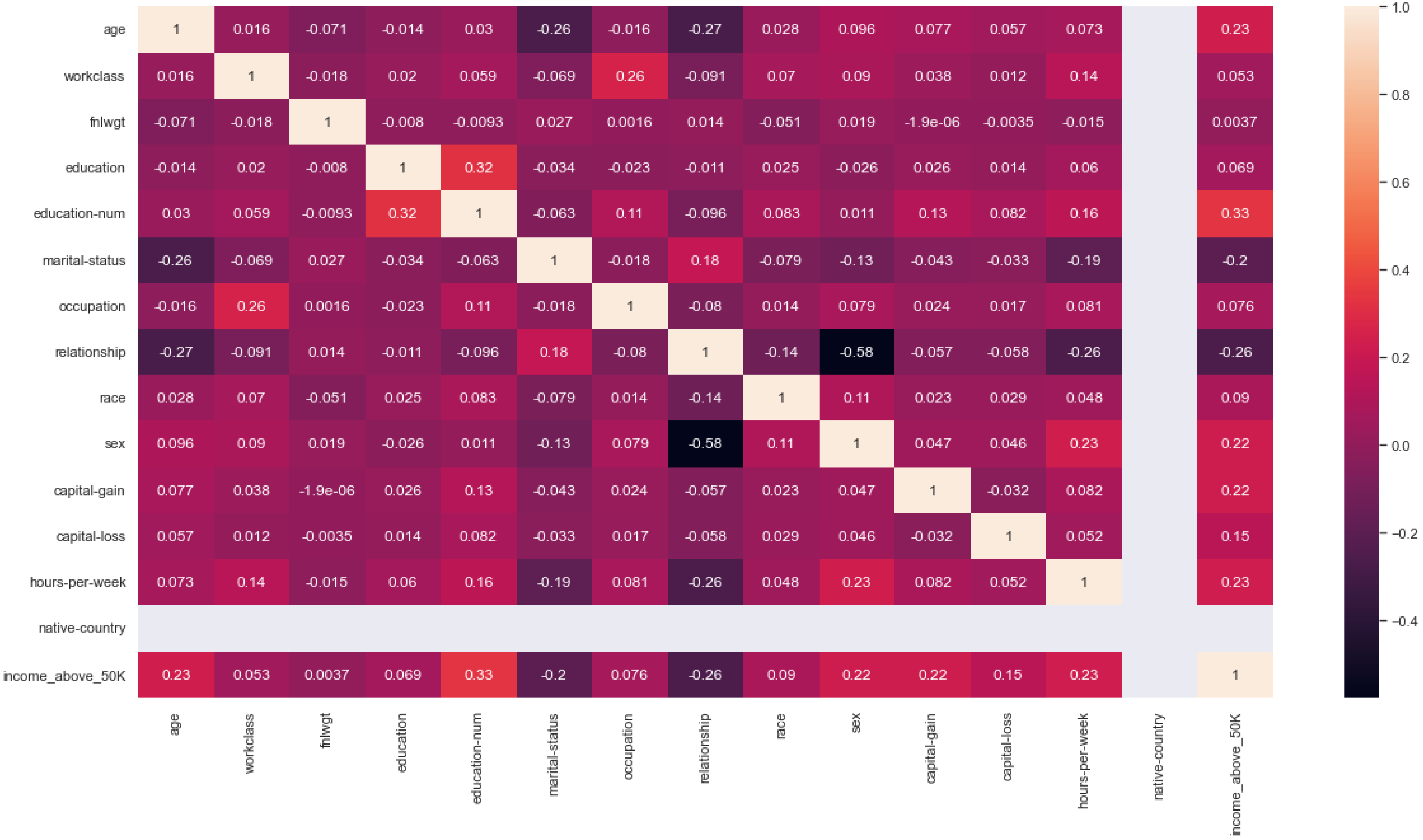
```python
census["workclass"] = LabelEncoder().fit_transform(census["workclass"])
census["education"] = LabelEncoder().fit_transform(census["education"])
census["marital-status"] = LabelEncoder().fit_transform(census["marital-status"])
census["occupation"] = LabelEncoder().fit_transform(census["occupation"])
census["relationship"] = LabelEncoder().fit_transform(census["relationship"])
census["race"] = LabelEncoder().fit_transform(census["race"])
census["sex"] = LabelEncoder().fit_transform(census["sex"])
census["native-country"] = LabelEncoder().fit_transform(census["native-country"])
```

```python
census.head()
```

Python

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income_above_50K
0	25	4	226802	1	7	4	7	3	2	1	0	0	40	0	0
1	38	4	89814	11	9	2	5	0	4	1	0	0	50	0	0
2	28	2	336951	7	12	2	11	0	4	1	0	0	40	0	1
3	44	4	160323	15	10	2	7	0	2	1	7688	0	40	0	1
4	18	0	103497	15	10	4	0	3	4	0	0	0	30	0	0

PREPROCESSING



DATA DIVISION

PREPROCESSING

```python
x = census.iloc[:, :-1]
y = census.iloc[:, -1]
x
```

Python

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	25	4	226802	1	7	4	7	3	2	1	0	0	40	0
1	38	4	89814	11	9	2	5	0	4	1	0	0	50	0
2	28	2	336951	7	12	2	11	0	4	1	0	0	40	0
3	44	4	160323	15	10	2	7	0	2	1	7688	0	40	0
4	18	0	103497	15	10	4	0	3	4	0	0	0	30	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
48837	27	4	257302	7	12	2	13	5	4	0	0	0	38	0
48838	40	4	154374	11	9	2	7	0	4	1	0	0	40	0
48839	58	4	151910	11	9	6	1	4	4	0	0	0	40	0
48840	22	4	201490	11	9	4	1	3	4	1	0	0	20	0
48841	52	5	287927	11	9	2	4	5	4	0	15024	0	40	0

43832 rows × 14 columns

PREPROCESSING

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
print(x_train.shape)
print(y_train.shape)
print(y_train.shape)
```

[155]

Python

```
(35065, 14)
(35065,)
(35065,)
```

APPLY ML MODEL

ML MODELS

- LOGISTIC REGRESSION
- SVM
- KNN
- Random Forest
- ANN

PREPROCESSING

Logistic Regression

```
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print('Accuracy Score: ', accuracy_score(y_pred, y_test))
print('Confusion Matrix: ', confusion_matrix(y_pred, y_test))
print('Classification Report', classification_report(y_test,y_pred))
```

57]

```
Accuracy Score:  0.7914908178396258
Confusion Matrix:  [[6367 1614]
 [ 214  572]]
Classification Report               precision    recall  f1-score   support

           0       0.80      0.97      0.87      6581
           1       0.73      0.26      0.38      2186

    accuracy                           0.79      8767
   macro avg       0.76      0.61      0.63      8767
weighted avg       0.78      0.79      0.75      8767
```

Support Vector Machine

```
model1 = SVC()
model1.fit(x_train, y_train)
y_pred = model1.predict(x_test)
print('Accuracy Score: ', accuracy_score(y_pred, y_test))
print('Confusion Matrix: ', confusion_matrix(y_pred, y_test))
print('Classification Report', classification_report(y_test,y_pred))
```

.59]

```
Accuracy Score:  0.7884110870309113
Confusion Matrix:  [[6571 1845]
 [  10  341]]
Classification Report               precision    recall  f1-score   support

           0       0.78      1.00      0.88      6581
           1       0.97      0.16      0.27      2186

    accuracy                           0.79      8767
   macro avg       0.88      0.58      0.57      8767
weighted avg       0.83      0.79      0.72      8767
```

PREPROCESSING

K-Nearest Neighbors

```
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(x_train, y_train)
y_pred = model2.predict(x_test)
print('Accuracy Score: ', accuracy_score(y_pred, y_test))
print('Confusion Matrix: ', confusion_matrix(y_pred, y_test))
print('Classification Report', classification_report(y_test,y_pred))
```

[161]

```
c:\Users\Lenovo\anaconda3\envs\streamlit\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarning:

Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along
SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is ta
be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.


Accuracy Score:  0.7731264970913654
Confusion Matrix:  [[6032 1440]
 [ 549  746]]
Classification Report               precision    recall  f1-score   support

           0       0.81      0.92      0.86      6581
           1       0.58      0.34      0.43      2186
```

PREPROCESSING

MLP Classifier

+ Code + Markdown

```
model3 = MLPClassifier(max_iter=100, alpha = 0.001, random_state=0)
model3.fit(x_train, y_train)
y_pred = model3.predict(x_test)
print('Accuracy Score: ', accuracy_score(y_pred, y_test))
print('Confusion Matrix: ', confusion_matrix(y_pred, y_test))
print('Classification Report', classification_report(y_test,y_pred))
```

[64]

```
Accuracy Score:  0.7902361126953348
Confusion Matrix:  [[6503 1761]
 [  78  425]]
Classification Report               precision    recall  f1-score   support

           0       0.79      0.99      0.88      6581
           1       0.84      0.19      0.32      2186

    accuracy                           0.79      8767
   macro avg       0.82      0.59      0.60      8767
weighted avg       0.80      0.79      0.74      8767
```

PREPROCESSING

Random Forest

+ Code + Markdown

```
model4 = RandomForestClassifier(n_estimators=50, max_depth=5, random_state=0)
model4.fit(x_train, y_train)
y_pred = model4.predict(x_test)
print('Accuracy Score: ', accuracy_score(y_pred, y_test))
print('Confusion Matrix: ', confusion_matrix(y_pred, y_test))
print('Classification Report', classification_report(y_test,y_pred))
```

[166]

```
Accuracy Score:  0.8485228698528573
Confusion Matrix:  [[6329 1076]
 [ 252 1110]]
Classification Report               precision    recall  f1-score   support

           0       0.85      0.96      0.91      6581
           1       0.81      0.51      0.63      2186

    accuracy                           0.85      8767
   macro avg       0.83      0.73      0.77      8767
weighted avg       0.84      0.85      0.84      8767
```

Thank You

24Slides