INT247 (MACHINE LEARNING FOUNDATION) ACADEMIC TASK 2

on

Census Income Report

Submitted by

Name: Satya Chandra Reddy Sabbella

Registration No: 11805549

Section and Roll No: KM007 – A16

Parent Section: K18GR

Programme Name: B.Tech - CSE

Under the Guidance of

Dr. Aditya Khamparia

School of Computer Science & Engineering

Lovely Professional University, Phagwara

Adult UCI Dataset (Census Income) Analysis with Python

Adult UCI dataset is one of the popular datasets for practice. It is a **Supervised binary classification problem**.

Aim is to predict whether a person makes over 50k a year

Details of the Dataset

The dataset contains a mix of categorical and numeric type data.

Categorical Attributes

- **workclass**: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
  - Individual work category
- **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.
  - Individual's highest education degree
- **marital-status**: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
  - Individual marital status
- **occupation**: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspect, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
  - Individual's occupation
- **relationship**: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
  - Individual's relation in a family
- **race**: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
  - Race of Individual
- **sex**: Female, Male.
- **native-country**: United-States, 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, Trinidad Tobago, Peru, Hong, Holland-Netherlands.
  - Individual's native country

Continuous Attributes

- **age**: continuous.
  - Age of an individual
- **fnlwgt**: final weight, continuous.

- The weights on the CPS files are controlled to independent estimates of the civilian noninstitutional population of the US. These are prepared monthly for us by Population Division here at the Census Bureau.
- **capital-gain**: continuous.
- **capital-loss**: continuous.
- **hours-per-week**: continuous.
    - Individual’s working hour per week

Sample of the dataset-

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	workclass	fnlwgt	education	educationa	marital-sta	occupatio	relationshi	race	gender	capital-gai	capital-los	hours-per-	native-cou	income
2	25	Private	226802	11th	7	Never-mai	Machine-c	Own-child	Black	Male	0	0	40	United-Sta	<=50K
3	38	Private	89814	HS-grad	9	Married-ci	Farming-fi	Husband	White	Male	0	0	50	United-Sta	<=50K
4	28	Local-gov	336951	Assoc-acd	12	Married-ci	Protective	Husband	White	Male	0	0	40	United-Sta	>50K
5	44	Private	160323	Some-coll	10	Married-ci	Machine-c	Husband	Black	Male	7688	0	40	United-Sta	>50K
6	18	?	103497	Some-coll	10	Never-mai	?	Own-child	White	Female	0	0	30	United-Sta	<=50K
7	34	Private	198693	10th	6	Never-mai	Other-serv	Not-in-fan	White	Male	0	0	30	United-Sta	<=50K
8	29	?	227026	HS-grad	9	Never-mai	?	Unmarried	Black	Male	0	0	40	United-Sta	<=50K
9	63	Self-emp-r	104626	Prof-schoo	15	Married-ci	Prof-speci	Husband	White	Male	3103	0	32	United-Sta	>50K
10	24	Private	369667	Some-coll	10	Never-mai	Other-serv	Unmarried	White	Female	0	0	40	United-Sta	<=50K
11	55	Private	104996	7th-8th	4	Married-ci	Craft-repa	Husband	White	Male	0	0	10	United-Sta	<=50K
12	65	Private	184454	HS-grad	9	Married-ci	Machine-c	Husband	White	Male	6418	0	40	United-Sta	>50K
13	36	Federal-go	212465	Bachelors	13	Married-ci	Adm-cleric	Husband	White	Male	0	0	40	United-Sta	<=50K
14	26	Private	82091	HS-grad	9	Never-mai	Adm-cleric	Not-in-fan	White	Female	0	0	39	United-Sta	<=50K
15	58	?	299831	HS-grad	9	Married-ci	?	Husband	White	Male	0	0	35	United-Sta	<=50K
16	48	Private	279724	HS-grad	9	Married-ci	Machine-c	Husband	White	Male	3103	0	48	United-Sta	>50K
17	43	Private	346189	Masters	14	Married-ci	Exec-mana	Husband	White	Male	0	0	50	United-Sta	>50K
18	20	State-gov	444554	Some-coll	10	Never-mai	Other-serv	Own-child	White	Male	0	0	25	United-Sta	<=50K
19	43	Private	128354	HS-grad	9	Married-ci	Adm-cleric	Wife	White	Female	0	0	30	United-Sta	<=50K
20	37	Private	60548	HS-grad	9	Widowed	Machine-c	Unmarried	White	Female	0	0	20	United-Sta	<=50K
21	40	Private	85019	Doctorate	16	Married-ci	Prof-speci	Husband	Asian-Pac-	Male	0	0	45	?	>50K
22	34	Private	107914	Bachelors	13	Married-ci	Tech-supp	Husband	White	Male	0	0	47	United-Sta	>50K
23	34	Private	238588	Some-coll	10	Never-mai	Other-serv	Own-child	Black	Female	0	0	35	United-Sta	<=50K
24	72	?	132015	7th-8th	4	Divorced	?	Not-in-fan	White	Female	0	0	6	United-Sta	<=50K
25	25	Private	220931	Bachelors	13	Never-mai	Prof-speci	Not-in-fan	White	Male	0	0	43	Peru	<=50K
26	25	Private	205947	Bachelors	13	Married-ci	Prof-speci	Husband	White	Male	0	0	40	United-Sta	<=50K
27	45	Self-emp-r	432824	HS-grad	9	Married-ci	Craft-repa	Husband	White	Male	7298	0	90	United-Sta	>50K

Loading basic libraries for the data-

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Exploring the data –

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
6	29	?	227026	HS-grad	9	Never-married	?	Unmarried	Black	Male	0	0	40	United-States	<=50K
7	63	Self-emp-not-inc	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband	White	Male	3103	0	32	United-States	>50K
8	24	Private	369667	Some-college	10	Never-married	Other-service	Unmarried	White	Female	0	0	40	United-States	<=50K
9	55	Private	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	10	United-States	<=50K

This dataset contains few unwanted values ('?') but it doesn't contain any null values in it but is filled with **question marks** as highlighted above. I have replaced the values with the mode values as shown below.

Understanding the data –

```
In [3]: df.shape
Out[3]: (48842, 15)
```

```
In [4]: df.dtypes
Out[4]: age                 int64
        workclass          object
        fnlwgt              int64
        education          object
        educational-num     int64
        marital-status     object
        occupation         object
        relationship       object
        race               object
        gender             object
        capital-gain        int64
        capital-loss        int64
        hours-per-week      int64
        native-country     object
        income             object
        dtype: object
```

```
In [5]: df.isnull().sum()
Out[5]: age                0
        workclass          0
        fnlwgt             0
        education          0
        educational-num    0
        marital-status     0
        occupation         0
        relationship       0
        race               0
        gender             0
        capital-gain       0
        capital-loss       0
        hours-per-week     0
        native-country     0
        income             0
        dtype: int64
```

This data contains 48,842 rows and 15 columns. This data type is a mixture of categorical and numerical data.

Counting the '?' values –

```
In [8]: df['workclass'].value_counts()

Out[8]: Private             33906
        Self-emp-not-inc     3862
        Local-gov            3136
        ?                    2799
        State-gov            1981
        Self-emp-inc         1695
        Federal-gov          1432
        Without-pay            21
        Never-worked           10
        Name: workclass, dtype: int64
```

```
In [9]: df['occupation'].value_counts()

Out[9]: Prof-specialty       6172
        Craft-repair         6112
        Exec-managerial      6086
        Adm-clerical         5611
        Sales                5504
        Other-service        4923
        Machine-op-inspct    3022
        ?                    2809
        Transport-moving     2355
        Handlers-cleaners    2072
        Farming-fishing      1490
        Tech-support         1446
        Protective-serv       983
        Priv-house-serv       242
        Armed-Forces           15
        Name: occupation, dtype: int64
```

```
In [10]: df['native-country'].value_counts()

Out[10]: United-States                43832
         Mexico                         951
         ?                              857
         Philippines                    295
         Germany                        206
         Puerto-Rico                    184
         Canada                         182
         El-Salvador                    155
         India                          151
         Cuba                           138
```

We can see that only three columns have '?' values and I cleaned it up using feature engineering.

Preparing Data –

```
In [16]: df['workclass'] = df['workclass'].replace('?', 'Private')
         df['occupation'] = df['occupation'].replace('?', 'Prof-specialty')
         df['native-country'] = df['native-country'].replace('?', 'United-State
         s')
```

```
In [17]: df.head(10)
```

Out[17]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationsh
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband
4	18	Private	103497	Some-college	10	Never-married	Prof-specialty	Own-child
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-fami
6	29	Private	227026	HS-grad	9	Never-married	Prof-specialty	Unmarried
7	63	Self-emp-not-inc	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband
8	24	Private	369667	Some-college	10	Never-married	Other-service	Unmarried

The '?' values have been handeled and the obtained new dataset I have cleaned furthur more usinf feature engineering.

Feature Engineering –

```
In [18]: df.education= df.education.replace(['Preschool', '1st-4th', '5th-6th',
         '7th-8th', '9th','10th', '11th', '12th'], 'School')
         df.education = df.education.replace('HS-grad', 'High school')
         df.education = df.education.replace(['Assoc-voc', 'Assoc-acdm', 'Prof-sc
         hool', 'Some-college'], 'High-Educ')
         df.education = df.education.replace('Bachelors', 'Undergrad')
         df.education = df.education.replace('Masters', 'Grad')
```

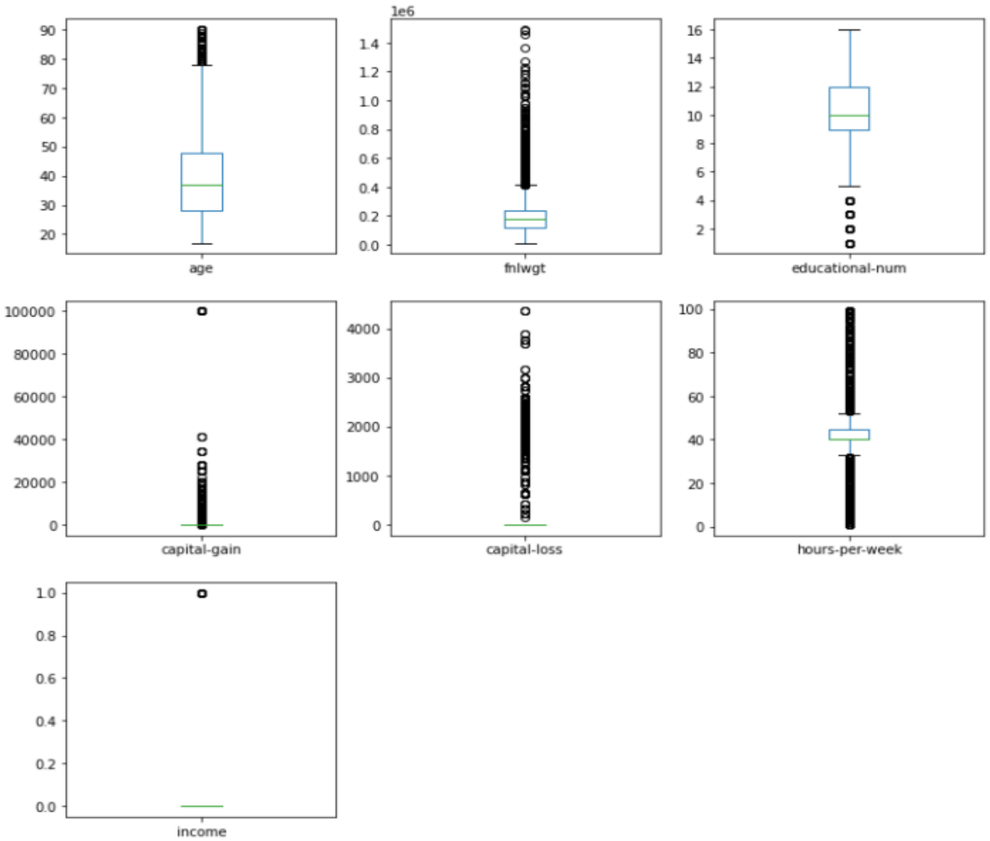
```
In [19]: df['marital-status']= df['marital-status'].replace(['Married-civ-spous
         e', 'Married-AF-spouse'], 'Married')
         df['marital-status']= df['marital-status'].replace(['Never-married'], 'N
         ot-married')
         df['marital-status']= df['marital-status'].replace(['Divorced', 'Separat
         ed','Widowed',
                                                               'Married-spouse-absen
         t'], 'other')
```

```
In [20]: df.income = df.income.replace('<=50K', 0)
         df.income = df.income.replace('>50K', 1)
```

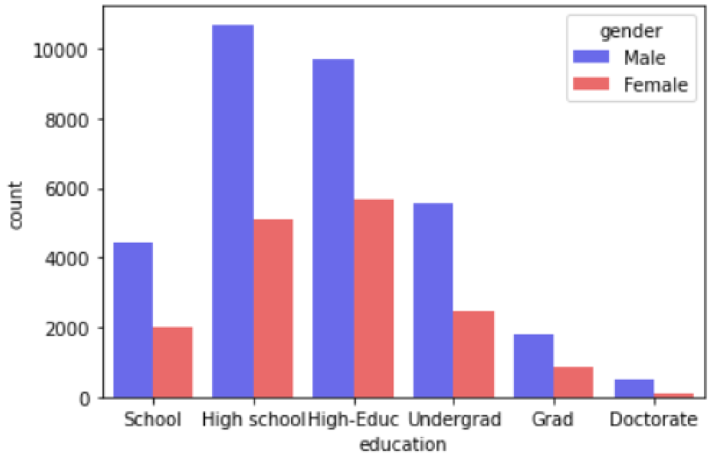
By doing this I've simplified the data in the rows more simpler than before.

Data Visualization –

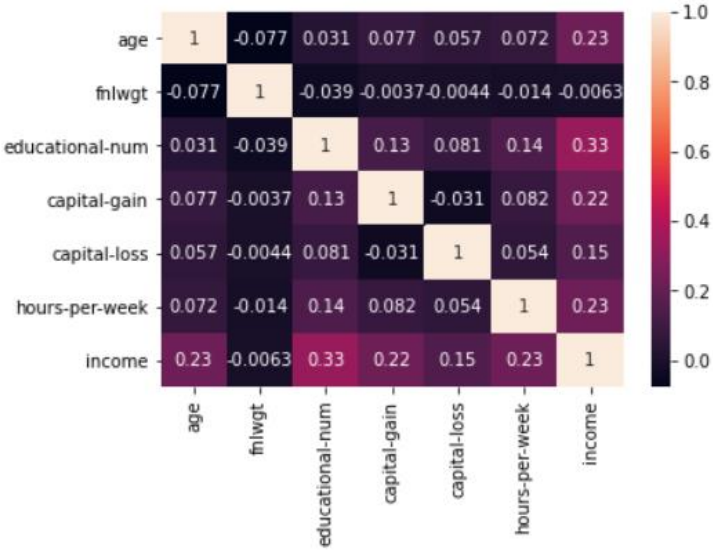
```
In [27]: df.plot(kind='box', figsize=(12,12), layout=(3,3), sharex=False, subplot
         s=True);
```



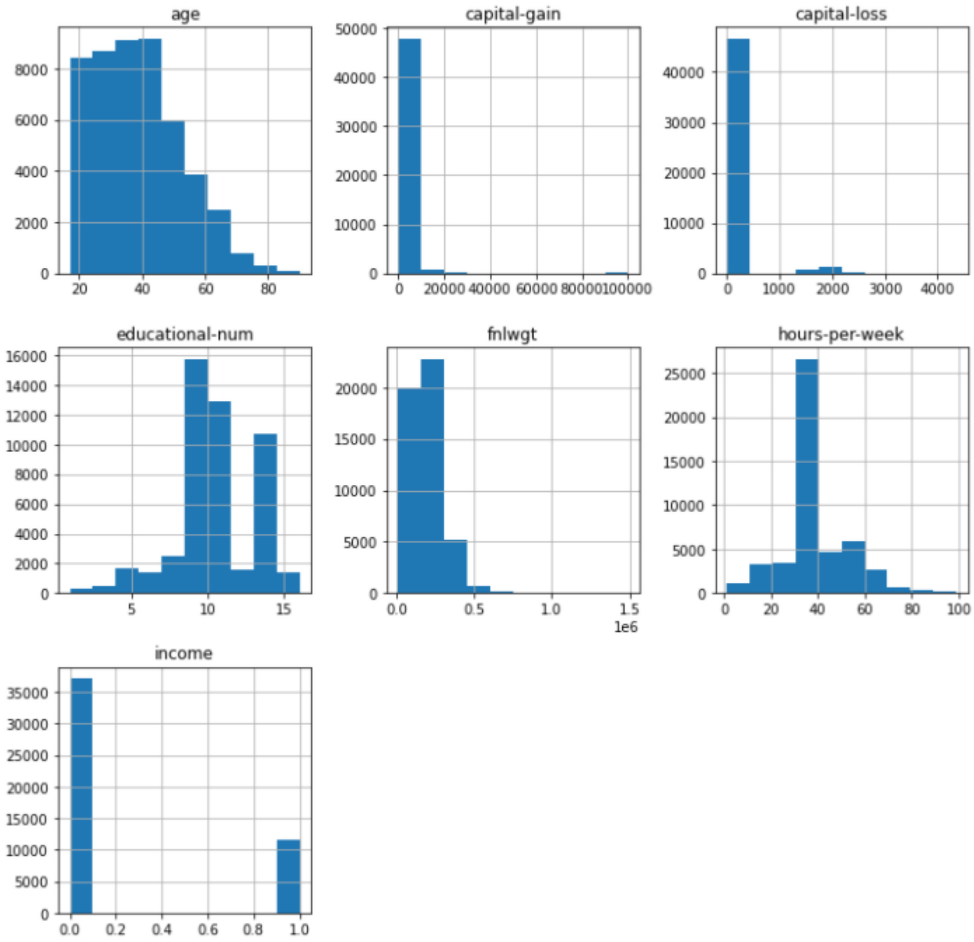
In [28]:
```
sns.countplot(df['education'], hue='gender', data=df, palette='seismic');
```



In [25]:
```
sns.heatmap(df.corr(), annot=True);
```



```
In [26]: df.hist(figsize=(12,12), layout=(3,3), sharex=False);
```



Key Findings

- The minimum age is 17 and the maximum is 90 years, most of the working age group lies between 20-40
- The minimum hours-per-week is 1 and maximum is 90, with most of the count lying between 30-40
- outliers observed in almost all the numeric features, these are the extreme values that are present in the data.
- Not very strong correlation observed among variables

Scaling the data-

In [29]:
```
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

In [30]:
```
df1= df.copy()
df1= df1.apply(LabelEncoder().fit_transform)
df1.head()
```

Out[30]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationshi
0	8	3	19329	4	6	1	6	3
1	21	3	4212	2	8	0	4	0
2	11	1	25340	3	11	0	10	0
3	27	3	11201	3	9	0	6	0
4	1	3	5411	3	9	1	9	3

In [31]:
```
ss= StandardScaler().fit(df1.drop('income', axis=1))
```

In [32]:
```
X= ss.transform(df1.drop('income', axis=1))
y= df['income']

X
```

Out[32]:
```
array([[-0.99512893, -0.08972675,  0.70632366, ..., -0.20508013,
        -0.03226706,  0.25969378],
       [-0.04694151, -0.08972675, -1.19627994, ..., -0.20508013,
         0.78106212,  0.25969378],
       [-0.77631645, -1.8902337 ,  1.46285937, ..., -0.20508013,
        -0.03226706,  0.25969378],
       ...,
       [ 1.41180837, -0.08972675, -0.42803939, ..., -0.20508013,
        -0.03226706,  0.25969378],
       [-1.21394141, -0.08972675,  0.38966357, ..., -0.20508013,
        -1.65892543,  0.25969378],
       [ 0.97418341,  0.81052673,  1.20875097, ..., -0.20508013,
        -0.03226706,  0.25969378]])
```

- Create X and y object to store the independent variable (X) and dependent variable(y).
- Perform Standard Scaling to scale the data
- Label Encoding is performed to convert the categorical data into numeric format
- Label Encoder makes the data suitable for machine
- Perform fit and Transform
- Split the dataset into train and test split

Hyperparameter Tuning –

In [33]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=40)
```

In [34]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

model_params = {
    'random_forest': {
        'model': RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10]
        }
    },
    'logistic_regression' : {
        'model': LogisticRegression(solver='liblinear',multi_class='aut
o'),
        'params': {
            'C': [1,5,10]
        }
    }
}
```

In [35]:
```
from sklearn.model_selection import GridSearchCV
scores = []

for model_name, mp in model_params.items():
    clf =  GridSearchCV(mp['model'], mp['params'], cv=10, return_train_s
core=False)
    clf.fit(X, y)
    scores.append({
        'model': model_name,
        'best_score': clf.best_score_,
        'best_params': clf.best_params_
    })

df2 = pd.DataFrame(scores,columns=['model','best_score','best_params'])
df2
```

Out[35]:

	model	best_score	best_params
0	random_forest	0.852361	{'n_estimators': 10}
1	logistic_regression	0.838172	{'C': 1}

Hyperparameter Tuning is done to find out the best parameters for a certain models to train data, it is done using GridSearchCV, it is a special module specifically used for Hyperparameter Tuning.

Choosing on 3 models and training them and reiterating –

In [36]:
```
lr = LogisticRegression(C=1)

model = lr.fit(X_train, y_train)
prediction = model.predict(X_test)

print("Acc on training data: {:,.3f}".format(lr.score(X_train, y_trai
n)))
print("Acc on test data: {:,.3f}".format(lr.score(X_test, y_test)))
```

```
Acc on training data: 0.838
Acc on test data: 0.839
```

In [37]:
```
rfc = RandomForestClassifier(n_estimators=10)

model1 = rfc.fit(X_train, y_train)
prediction1 = model1.predict(X_test)

print("Acc on training data: {:,.3f}".format(rfc.score(X_train, y_trai
n)))
print("Acc on test data: {:,.3f}".format(rfc.score(X_test, y_test)))
```

```
Acc on training data: 0.989
Acc on test data: 0.851
```

In [38]:
```
dtc = DecisionTreeClassifier()
model2 = dtc.fit(X_train, y_train)
prediction2 = model2.predict(X_test)
accuracy_score(y_test, prediction2)

print("Accuracy on training set: {:.3f}".format(dtc.score(X_train, y_tra
in)))
print("Accuracy on test set: {:.3f}".format(dtc.score(X_test, y_test)))
```

```
Accuracy on training set: 1.000
Accuracy on test set: 0.814
```

Using Hyperparameter Tuning I have set the parameters for the models I have chose to train my data and I have use 3 models

- Logistic Regression
- Random Forest Classifier
- Decision Tree Classifier

By training the module and predicting their accuracy I've concluded that Random Forest Classifier is a better Machine Learning model than the rest two because the accuracy on training it gave me is 85%.

Confusion Matrix (Evaluation / Highest accuracy)–

In [39]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [40]:
```
print(confusion_matrix(y_test, prediction))
```

```
[[10380   758]
 [ 1608  1907]]
```

In [41]:
```
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

           0       0.87      0.93      0.90     11138
           1       0.72      0.54      0.62      3515

    accuracy                           0.84     14653
   macro avg       0.79      0.74      0.76     14653
weighted avg       0.83      0.84      0.83     14653
```

In [42]:
```
print(confusion_matrix(y_test, prediction1))
print(classification_report(y_test, prediction1))
```

```
[[10379   759]
 [ 1424  2091]]
              precision    recall  f1-score   support

           0       0.88      0.93      0.90     11138
           1       0.73      0.59      0.66      3515

    accuracy                           0.85     14653
   macro avg       0.81      0.76      0.78     14653
weighted avg       0.84      0.85      0.85     14653
```

In [43]:
```
print(confusion_matrix(y_test, prediction2))
print(classification_report(y_test, prediction2))
```

```
[[9709 1429]
 [1297 2218]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.88     11138
           1       0.61      0.63      0.62      3515

    accuracy                           0.81     14653
   macro avg       0.75      0.75      0.75     14653
weighted avg       0.82      0.81      0.82     14653
```