Census_Income_Classification_SVC

November 15, 2022

**

Classificaton Problem (SVC, Logistic, SVM Kernel

* Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset**

Import required libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve,␣
 ↪roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
%matplotlib inline
```

Complete dataset is available on my GitHub * GitHub Link: https://github.com/subhashdixit/Support_Vector_Machines/tree/main/SVC/Census_Income_Classification

Read Data From GitHub

[2]:
```
url_train = 'adult_data.csv'
url_test = 'adult_test.csv'
df_train = pd.read_csv(url_train, header = None)
df_test = pd.read_csv(url_test, header = None, skiprows = 1)
```

[3]:
```
df_train
```

[3]:
```
        0                 1       2            3   4   \
0      39         State-gov   77516    Bachelors  13
1      50  Self-emp-not-inc   83311    Bachelors  13
2      38           Private  215646      HS-grad   9
```

3	53	Private	234721	11th	7
4	28	Private	338409	Bachelors	13
5	37	Private	284582	Masters	14
6	49	Private	160187	9th	5
7	52	Self-emp-not-inc	209642	HS-grad	9
8	31	Private	45781	Masters	14
9	42	Private	159449	Bachelors	13
10	37	Private	280464	Some-college	10
11	30	State-gov	141297	Bachelors	13
12	23	Private	122272	Bachelors	13
13	32	Private	205019	Assoc-acdm	12
14	40	Private	121772	Assoc-voc	11
15	34	Private	245487	7th-8th	4
16	25	Self-emp-not-inc	176756	HS-grad	9
17	32	Private	186824	HS-grad	9
18	38	Private	28887	11th	7
19	43	Self-emp-not-inc	292175	Masters	14
20	40	Private	193524	Doctorate	16
21	54	Private	302146	HS-grad	9
22	35	Federal-gov	76845	9th	5
23	43	Private	117037	11th	7
24	59	Private	109015	HS-grad	9
25	56	Local-gov	216851	Bachelors	13
26	19	Private	168294	HS-grad	9
27	54	?	180211	Some-college	10
28	39	Private	367260	HS-grad	9
29	49	Private	193366	HS-grad	9
…	..	…	…	…	..
32531	30	?	33811	Bachelors	13
32532	34	Private	204461	Doctorate	16
32533	54	Private	337992	Bachelors	13
32534	37	Private	179137	Some-college	10
32535	22	Private	325033	12th	8
32536	34	Private	160216	Bachelors	13
32537	30	Private	345898	HS-grad	9
32538	38	Private	139180	Bachelors	13
32539	71	?	287372	Doctorate	16
32540	45	State-gov	252208	HS-grad	9
32541	41	?	202822	HS-grad	9
32542	72	?	129912	HS-grad	9
32543	45	Local-gov	119199	Assoc-acdm	12
32544	31	Private	199655	Masters	14
32545	39	Local-gov	111499	Assoc-acdm	12
32546	37	Private	198216	Assoc-acdm	12
32547	43	Private	260761	HS-grad	9
32548	65	Self-emp-not-inc	99359	Prof-school	15
32549	43	State-gov	255835	Some-college	10

32550	43	Self-emp-not-inc	27242	Some-college	10
32551	32	Private	34066	10th	6
32552	43	Private	84661	Assoc-voc	11
32553	32	Private	116138	Masters	14
32554	53	Private	321865	Masters	14
32555	22	Private	310152	Some-college	10
32556	27	Private	257302	Assoc-acdm	12
32557	40	Private	154374	HS-grad	9
32558	58	Private	151910	HS-grad	9
32559	22	Private	201490	HS-grad	9
32560	52	Self-emp-inc	287927	HS-grad	9

	5	6	7 \
0	Never-married	Adm-clerical	Not-in-family
1	Married-civ-spouse	Exec-managerial	Husband
2	Divorced	Handlers-cleaners	Not-in-family
3	Married-civ-spouse	Handlers-cleaners	Husband
4	Married-civ-spouse	Prof-specialty	Wife
5	Married-civ-spouse	Exec-managerial	Wife
6	Married-spouse-absent	Other-service	Not-in-family
7	Married-civ-spouse	Exec-managerial	Husband
8	Never-married	Prof-specialty	Not-in-family
9	Married-civ-spouse	Exec-managerial	Husband
10	Married-civ-spouse	Exec-managerial	Husband
11	Married-civ-spouse	Prof-specialty	Husband
12	Never-married	Adm-clerical	Own-child
13	Never-married	Sales	Not-in-family
14	Married-civ-spouse	Craft-repair	Husband
15	Married-civ-spouse	Transport-moving	Husband
16	Never-married	Farming-fishing	Own-child
17	Never-married	Machine-op-inspct	Unmarried
18	Married-civ-spouse	Sales	Husband
19	Divorced	Exec-managerial	Unmarried
20	Married-civ-spouse	Prof-specialty	Husband
21	Separated	Other-service	Unmarried
22	Married-civ-spouse	Farming-fishing	Husband
23	Married-civ-spouse	Transport-moving	Husband
24	Divorced	Tech-support	Unmarried
25	Married-civ-spouse	Tech-support	Husband
26	Never-married	Craft-repair	Own-child
27	Married-civ-spouse	?	Husband
28	Divorced	Exec-managerial	Not-in-family
29	Married-civ-spouse	Craft-repair	Husband
…	…	…	…
32531	Never-married	?	Not-in-family
32532	Married-civ-spouse	Prof-specialty	Husband
32533	Married-civ-spouse	Exec-managerial	Husband

32534	Divorced	Adm-clerical	Unmarried
32535	Never-married	Protective-serv	Own-child
32536	Never-married	Exec-managerial	Not-in-family
32537	Never-married	Craft-repair	Not-in-family
32538	Divorced	Prof-specialty	Unmarried
32539	Married-civ-spouse	?	Husband
32540	Separated	Adm-clerical	Own-child
32541	Separated	?	Not-in-family
32542	Married-civ-spouse	?	Husband
32543	Divorced	Prof-specialty	Unmarried
32544	Divorced	Other-service	Not-in-family
32545	Married-civ-spouse	Adm-clerical	Wife
32546	Divorced	Tech-support	Not-in-family
32547	Married-civ-spouse	Machine-op-inspct	Husband
32548	Never-married	Prof-specialty	Not-in-family
32549	Divorced	Adm-clerical	Other-relative
32550	Married-civ-spouse	Craft-repair	Husband
32551	Married-civ-spouse	Handlers-cleaners	Husband
32552	Married-civ-spouse	Sales	Husband
32553	Never-married	Tech-support	Not-in-family
32554	Married-civ-spouse	Exec-managerial	Husband
32555	Never-married	Protective-serv	Not-in-family
32556	Married-civ-spouse	Tech-support	Wife
32557	Married-civ-spouse	Machine-op-inspct	Husband
32558	Widowed	Adm-clerical	Unmarried
32559	Never-married	Adm-clerical	Own-child
32560	Married-civ-spouse	Exec-managerial	Wife

	8	9	10	11	12	13	14
0	White	Male	2174	0	40	United-States	<=50K
1	White	Male	0	0	13	United-States	<=50K
2	White	Male	0	0	40	United-States	<=50K
3	Black	Male	0	0	40	United-States	<=50K
4	Black	Female	0	0	40	Cuba	<=50K
5	White	Female	0	0	40	United-States	<=50K
6	Black	Female	0	0	16	Jamaica	<=50K
7	White	Male	0	0	45	United-States	>50K
8	White	Female	14084	0	50	United-States	>50K
9	White	Male	5178	0	40	United-States	>50K
10	Black	Male	0	0	80	United-States	>50K
11	Asian-Pac-Islander	Male	0	0	40	India	>50K
12	White	Female	0	0	30	United-States	<=50K
13	Black	Male	0	0	50	United-States	<=50K
14	Asian-Pac-Islander	Male	0	0	40	?	>50K
15	Amer-Indian-Eskimo	Male	0	0	45	Mexico	<=50K
16	White	Male	0	0	35	United-States	<=50K
17	White	Male	0	0	40	United-States	<=50K

```
18                  White    Male      0     0  50  United-States   <=50K
19                  White  Female      0     0  45  United-States    >50K
20                  White    Male      0     0  60  United-States    >50K
21                  Black  Female      0     0  20  United-States   <=50K
22                  Black    Male      0     0  40  United-States   <=50K
23                  White    Male      0  2042  40  United-States   <=50K
24                  White  Female      0     0  40  United-States   <=50K
25                  White    Male      0     0  40  United-States    >50K
26                  White    Male      0     0  40  United-States   <=50K
27     Asian-Pac-Islander    Male      0     0  60          South    >50K
28                  White    Male      0     0  80  United-States   <=50K
29                  White    Male      0     0  40  United-States   <=50K
…                       …       …      …     …  ..            …     …
32531  Asian-Pac-Islander  Female      0     0  99  United-States   <=50K
32532               White    Male      0     0  60  United-States    >50K
32533  Asian-Pac-Islander    Male      0     0  50          Japan    >50K
32534               White  Female      0     0  39  United-States   <=50K
32535               Black    Male      0     0  35  United-States   <=50K
32536               White  Female      0     0  55  United-States    >50K
32537               Black    Male      0     0  46  United-States   <=50K
32538               Black  Female  15020     0  45  United-States    >50K
32539               White    Male      0     0  10  United-States    >50K
32540               White  Female      0     0  40  United-States   <=50K
32541               Black  Female      0     0  32  United-States   <=50K
32542               White    Male      0     0  25  United-States   <=50K
32543               White  Female      0     0  48  United-States   <=50K
32544               Other  Female      0     0  30  United-States   <=50K
32545               White  Female      0     0  20  United-States    >50K
32546               White  Female      0     0  40  United-States   <=50K
32547               White    Male      0     0  40         Mexico   <=50K
32548               White    Male   1086     0  60  United-States   <=50K
32549               White  Female      0     0  40  United-States   <=50K
32550               White    Male      0     0  50  United-States   <=50K
32551  Amer-Indian-Eskimo    Male      0     0  40  United-States   <=50K
32552               White    Male      0     0  45  United-States   <=50K
32553  Asian-Pac-Islander    Male      0     0  11         Taiwan   <=50K
32554               White    Male      0     0  40  United-States    >50K
32555               White    Male      0     0  40  United-States   <=50K
32556               White  Female      0     0  38  United-States   <=50K
32557               White    Male      0     0  40  United-States    >50K
32558               White  Female      0     0  40  United-States   <=50K
32559               White    Male      0     0  20  United-States   <=50K
32560               White  Female  15024     0  40  United-States    >50K

[32561 rows x 15 columns]
```

[4]:
```
df_test
```

[4]:

	0	1	2	3	4	5 \
0	25	Private	226802	11th	7	Never-married
1	38	Private	89814	HS-grad	9	Married-civ-spouse
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse
3	44	Private	160323	Some-college	10	Married-civ-spouse
4	18	?	103497	Some-college	10	Never-married
5	34	Private	198693	10th	6	Never-married
6	29	?	227026	HS-grad	9	Never-married
7	63	Self-emp-not-inc	104626	Prof-school	15	Married-civ-spouse
8	24	Private	369667	Some-college	10	Never-married
9	55	Private	104996	7th-8th	4	Married-civ-spouse
10	65	Private	184454	HS-grad	9	Married-civ-spouse
11	36	Federal-gov	212465	Bachelors	13	Married-civ-spouse
12	26	Private	82091	HS-grad	9	Never-married
13	58	?	299831	HS-grad	9	Married-civ-spouse
14	48	Private	279724	HS-grad	9	Married-civ-spouse
15	43	Private	346189	Masters	14	Married-civ-spouse
16	20	State-gov	444554	Some-college	10	Never-married
17	43	Private	128354	HS-grad	9	Married-civ-spouse
18	37	Private	60548	HS-grad	9	Widowed
19	40	Private	85019	Doctorate	16	Married-civ-spouse
20	34	Private	107914	Bachelors	13	Married-civ-spouse
21	34	Private	238588	Some-college	10	Never-married
22	72	?	132015	7th-8th	4	Divorced
23	25	Private	220931	Bachelors	13	Never-married
24	25	Private	205947	Bachelors	13	Married-civ-spouse
25	45	Self-emp-not-inc	432824	HS-grad	9	Married-civ-spouse
26	22	Private	236427	HS-grad	9	Never-married
27	23	Private	134446	HS-grad	9	Separated
28	54	Private	99516	HS-grad	9	Married-civ-spouse
29	32	Self-emp-not-inc	109282	Some-college	10	Never-married
…	..	…	…	…	..	…
16251	81	?	26711	Assoc-voc	11	Married-civ-spouse
16252	60	Private	117909	Assoc-voc	11	Married-civ-spouse
16253	39	Private	229647	Bachelors	13	Never-married
16254	38	Private	149347	Masters	14	Married-civ-spouse
16255	43	Local-gov	23157	Masters	14	Married-civ-spouse
16256	23	Private	93977	HS-grad	9	Never-married
16257	73	Self-emp-inc	159691	Some-college	10	Divorced
16258	35	Private	176967	Some-college	10	Married-civ-spouse
16259	66	Private	344436	HS-grad	9	Widowed
16260	27	Private	430340	Some-college	10	Never-married
16261	40	Private	202168	Prof-school	15	Married-civ-spouse
16262	51	Private	82720	HS-grad	9	Married-civ-spouse
16263	22	Private	269623	Some-college	10	Never-married
16264	64	Self-emp-not-inc	136405	HS-grad	9	Widowed
16265	50	Local-gov	139347	Masters	14	Married-civ-spouse

16266	55	Private	224655	HS-grad	9	Separated
16267	38	Private	247547	Assoc-voc	11	Never-married
16268	58	Private	292710	Assoc-acdm	12	Divorced
16269	32	Private	173449	HS-grad	9	Married-civ-spouse
16270	48	Private	285570	HS-grad	9	Married-civ-spouse
16271	61	Private	89686	HS-grad	9	Married-civ-spouse
16272	31	Private	440129	HS-grad	9	Married-civ-spouse
16273	25	Private	350977	HS-grad	9	Never-married
16274	48	Local-gov	349230	Masters	14	Divorced
16275	33	Private	245211	Bachelors	13	Never-married
16276	39	Private	215419	Bachelors	13	Divorced
16277	64	?	321403	HS-grad	9	Widowed
16278	38	Private	374983	Bachelors	13	Married-civ-spouse
16279	44	Private	83891	Bachelors	13	Divorced
16280	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse

	6	7	8	9	\
0	Machine-op-inspct	Own-child	Black	Male	
1	Farming-fishing	Husband	White	Male	
2	Protective-serv	Husband	White	Male	
3	Machine-op-inspct	Husband	Black	Male	
4	?	Own-child	White	Female	
5	Other-service	Not-in-family	White	Male	
6	?	Unmarried	Black	Male	
7	Prof-specialty	Husband	White	Male	
8	Other-service	Unmarried	White	Female	
9	Craft-repair	Husband	White	Male	
10	Machine-op-inspct	Husband	White	Male	
11	Adm-clerical	Husband	White	Male	
12	Adm-clerical	Not-in-family	White	Female	
13	?	Husband	White	Male	
14	Machine-op-inspct	Husband	White	Male	
15	Exec-managerial	Husband	White	Male	
16	Other-service	Own-child	White	Male	
17	Adm-clerical	Wife	White	Female	
18	Machine-op-inspct	Unmarried	White	Female	
19	Prof-specialty	Husband	Asian-Pac-Islander	Male	
20	Tech-support	Husband	White	Male	
21	Other-service	Own-child	Black	Female	
22	?	Not-in-family	White	Female	
23	Prof-specialty	Not-in-family	White	Male	
24	Prof-specialty	Husband	White	Male	
25	Craft-repair	Husband	White	Male	
26	Adm-clerical	Own-child	White	Male	
27	Machine-op-inspct	Unmarried	Black	Male	
28	Craft-repair	Husband	White	Male	
29	Prof-specialty	Not-in-family	White	Male	

…	…	…	… …	
16251	?	Husband	White	Male
16252	Prof-specialty	Husband	White	Male
16253	Tech-support	Not-in-family	White	Female
16254	Prof-specialty	Husband	White	Male
16255	Exec-managerial	Husband	White	Male
16256	Machine-op-inspct	Own-child	White	Male
16257	Exec-managerial	Not-in-family	White	Female
16258	Protective-serv	Husband	White	Male
16259	Sales	Other-relative	White	Female
16260	Sales	Not-in-family	White	Female
16261	Prof-specialty	Husband	White	Male
16262	Craft-repair	Husband	White	Male
16263	Craft-repair	Own-child	White	Male
16264	Farming-fishing	Not-in-family	White	Male
16265	Prof-specialty	Wife	White	Female
16266	Priv-house-serv	Not-in-family	White	Female
16267	Adm-clerical	Unmarried	Black	Female
16268	Prof-specialty	Not-in-family	White	Male
16269	Handlers-cleaners	Husband	White	Male
16270	Adm-clerical	Husband	White	Male
16271	Sales	Husband	White	Male
16272	Craft-repair	Husband	White	Male
16273	Other-service	Own-child	White	Female
16274	Other-service	Not-in-family	White	Male
16275	Prof-specialty	Own-child	White	Male
16276	Prof-specialty	Not-in-family	White	Female
16277	?	Other-relative	Black	Male
16278	Prof-specialty	Husband	White	Male
16279	Adm-clerical	Own-child	Asian-Pac-Islander	Male
16280	Exec-managerial	Husband	White	Male

	10	11	12	13	14
0	0	0	40	United-States	<=50K.
1	0	0	50	United-States	<=50K.
2	0	0	40	United-States	>50K.
3	7688	0	40	United-States	>50K.
4	0	0	30	United-States	<=50K.
5	0	0	30	United-States	<=50K.
6	0	0	40	United-States	<=50K.
7	3103	0	32	United-States	>50K.
8	0	0	40	United-States	<=50K.
9	0	0	10	United-States	<=50K.
10	6418	0	40	United-States	>50K.
11	0	0	40	United-States	<=50K.
12	0	0	39	United-States	<=50K.
13	0	0	35	United-States	<=50K.

14	3103	0	48	United-States	>50K.
15	0	0	50	United-States	>50K.
16	0	0	25	United-States	<=50K.
17	0	0	30	United-States	<=50K.
18	0	0	20	United-States	<=50K.
19	0	0	45	?	>50K.
20	0	0	47	United-States	>50K.
21	0	0	35	United-States	<=50K.
22	0	0	6	United-States	<=50K.
23	0	0	43	Peru	<=50K.
24	0	0	40	United-States	<=50K.
25	7298	0	90	United-States	>50K.
26	0	0	20	United-States	<=50K.
27	0	0	54	United-States	<=50K.
28	0	0	35	United-States	<=50K.
29	0	0	60	United-States	<=50K.
…	… …	..		… …	
16251	2936	0	20	United-States	<=50K.
16252	7688	0	40	United-States	>50K.
16253	0	1669	40	United-States	<=50K.
16254	0	0	50	United-States	>50K.
16255	0	1902	50	United-States	>50K.
16256	0	0	40	United-States	<=50K.
16257	0	0	40	United-States	<=50K.
16258	0	0	40	United-States	<=50K.
16259	0	0	8	United-States	<=50K.
16260	0	0	45	United-States	<=50K.
16261	15024	0	55	United-States	>50K.
16262	0	0	40	United-States	<=50K.
16263	0	0	40	United-States	<=50K.
16264	0	0	32	United-States	<=50K.
16265	0	0	40	?	>50K.
16266	0	0	32	United-States	<=50K.
16267	0	0	40	United-States	<=50K.
16268	0	0	36	United-States	<=50K.
16269	0	0	40	United-States	<=50K.
16270	0	0	40	United-States	<=50K.
16271	0	0	48	United-States	<=50K.
16272	0	0	40	United-States	<=50K.
16273	0	0	40	United-States	<=50K.
16274	0	0	40	United-States	<=50K.
16275	0	0	40	United-States	<=50K.
16276	0	0	36	United-States	<=50K.
16277	0	0	40	United-States	<=50K.
16278	0	0	50	United-States	<=50K.
16279	5455	0	40	United-States	<=50K.
16280	0	0	60	United-States	>50K.

```
[16281 rows x 15 columns]
```

Data Set Information:

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Prediction task is to determine whether a person makes over 50K a year.

Attribute Information:

Listing of attributes:

50K, <=50K.

1. age: continuous
2. workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, 3. State-gov, Without-pay, Never-worked.
3. fnlwgt: continuous.
4. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, 6. Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5. education-num: continuous.
6. marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. 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.
8. relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. sex: Female, Male.
11. capital-gain: continuous.
12. capital-loss: continuous.
13. hours-per-week: continuous.
14. 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, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Merge test and train data set to perform EDA

[5]:
```
df = pd.concat([df_train,df_test])
```

[6]:
```
df1 = df.copy()
df = df.sample(n=10000)
```

[7]:
```
df.head()
```

[7]:
```
            0                  1       2             3   4                    5  \
10837  49                  ?  202874       HS-grad   9            Separated
8294   50        Federal-gov  183611  Some-college  10        Never-married
23745  48            Private  431513          10th   6   Married-civ-spouse
12888  43   Self-emp-not-inc  175943  Some-college  10        Never-married
24512  28            Private  177955          11th   7   Married-civ-spouse

                   6               7       8       9  10  11  12  \
10837              ?       Unmarried   White  Female   0   0  40
8294    Adm-clerical   Not-in-family   White    Male   0   0  40
23745   Craft-repair         Husband   White    Male   0   0  65
12888   Craft-repair   Not-in-family   White  Female   0   0  14
24512   Adm-clerical            Wife   White  Female   0   0  40

                   13        14
10837        Columbia     <=50K
8294    United-States     <=50K
23745   United-States      >50K
12888   United-States    <=50K.
24512          Mexico     <=50K
```

0.1 EDA

Rename the columns as per given description

[8]:
```python
rename_columns = {0 : 'age', 1 : 'workclass', 2 : 'fnlwgt', 3 : 'education', 4 :
 ↪ 'education-num', 5 : 'marital-status', 6 : 'occupation',
                  7 : 'relationship', 8 : 'race', 9 : 'sex', 10 :␣
 ↪'capital-gain', 11 : 'capital-loss', 12 : 'hours-per-week',
                  13 : 'native-country', 14 : 'class'}
df.rename(columns = rename_columns, inplace = True)
```

Information about the dataset

[9]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 10837 to 28389
Data columns (total 15 columns):
age                 10000 non-null int64
workclass           10000 non-null object
fnlwgt              10000 non-null int64
education           10000 non-null object
education-num       10000 non-null int64
marital-status      10000 non-null object
occupation          10000 non-null object
relationship        10000 non-null object
```

```
race              10000 non-null object
sex               10000 non-null object
capital-gain      10000 non-null int64
capital-loss      10000 non-null int64
hours-per-week    10000 non-null int64
native-country    10000 non-null object
class             10000 non-null object
dtypes: int64(6), object(9)
memory usage: 1.2+ MB
```

All the columns in the dataset

[10]:
```
df.columns
```

[10]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'class'],
      dtype='object')
```

Check unique values in each column

[11]:
```
for i in df.columns:
 ␣
 ↪print("--------------------------------------------------------------------------------")
  print(f"{i} : {df[i].unique()}")
 ␣
 ↪print("--------------------------------------------------------------------------------")
```

```
--------------------------------------------------------------------------------
age : [49 50 48 43 28 25 47 41 39 42 26 32 20 34 21 57 40 46 23 38 29 36 22 37
 45 30 62 27 33 18 64 44 54 19 61 55 17 31 24 35 56 59 60 67 52 53 65 51
 73 63 58 68 83 69 66 80 74 72 70 77 90 75 87 78 71 79 76 84 81 88 85 82]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
workclass : [' ?' ' Federal-gov' ' Private' ' Self-emp-not-inc' ' State-gov'
 ' Local-gov' ' Self-emp-inc' ' Never-worked' ' Without-pay']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
fnlwgt : [202874 183611 431513 … 353358 213385 206297]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
education : [' HS-grad' ' Some-college' ' 10th' ' 11th' ' Masters' ' 12th'
 ' Bachelors' ' Assoc-voc' ' Doctorate' ' Prof-school' ' Assoc-acdm'
 ' 7th-8th' ' 9th' ' 1st-4th' ' 5th-6th' ' Preschool']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
education-num : [ 9 10  6  7 14  8 13 11 16 15 12  4  5  2  3  1]
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
marital-status : [' Separated' ' Never-married' ' Married-civ-spouse'
 ' Married-spouse-absent' ' Divorced' ' Widowed' ' Married-AF-spouse']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
occupation : [' ?' ' Adm-clerical' ' Craft-repair' ' Prof-specialty' ' Other-
service'
 ' Exec-managerial' ' Handlers-cleaners' ' Tech-support'
 ' Transport-moving' ' Machine-op-inspct' ' Sales' ' Farming-fishing'
 ' Protective-serv' ' Priv-house-serv' ' Armed-Forces']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
relationship : [' Unmarried' ' Not-in-family' ' Husband' ' Wife' ' Own-child'
 ' Other-relative']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
race : [' White' ' Black' ' Other' ' Asian-Pac-Islander' ' Amer-Indian-Eskimo']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
sex : [' Female' ' Male']
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
capital-gain : [    0 15024  7688  4650  1055  5178   594  4787  6849 14344
3456  4386
  4064 10520 14084  2977  2597  7298  3942 99999  3325  3137  3103  8614
 15020  3908  1455  4865  2829  2346  2580  6418 10605  1151 27828  6497
  2885  3411 25124  2176  7978  3674  5013  2993  2174  4508  3781  2961
 25236 13550 20051  4687  2414  4416  6097  1506  2635  1409  2228  2407
  3464  1848   401  2653  9386  3418  1639   114  5556  2105  5060  4101
  7443  2907  2202  2290  1831  2964  3432  4931  1424  2354 10566  3887
 41310  3818  5455  2062  2036  6360 15831  3471  2329  6723  2009  1797
   914]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
capital-loss : [   0 1902 1974 1669 1564 1848 1977 1887 1340 2002 1590  880 1668
1380
 1617 1602 2051 2339 1740 2267 2042  625 1408 1485 1876 3900 1721 1672
 2057 1741  213 2377 2415 2179 2559 2258 1651 1719 2001 1573 1510 2603
 2174 1980 1579 1870 2392 2444 2754 1628 2824 1762 1411 2547 1504 3770
 2149 2246 2467 1258 1092 2129 1429 2163 1138 2206  419  653 4356 1726
 2457 2205 1825]
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
hours-per-week : [40 65 14 50 24 44 48 15 70 45 75 60 35 30 36 10 55 20 32 25  8
37 12 16
 43 38 46 63 96 56 17 31 72  3 52 85 47 18 42 66 39 80 33  4 34 53 99  7
 28 21 64  5  6 11  2 22 27 51 90 84 54  9 49 41 68 78 98 59 61 58 19 23
 13  1 81 26 67 57 62 94 74 73 77 76 88 29 89 86 92]
```

```
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
native-country : [' Columbia' ' United-States' ' Mexico' ' ?' ' China' '
Nicaragua'
 ' Philippines' ' Dominican-Republic' ' Germany' ' Japan' ' Cuba'
 ' Poland' ' Canada' ' Vietnam' ' Puerto-Rico' ' Ireland' ' France'
 ' Italy' ' Taiwan' ' El-Salvador' ' India' ' Peru' ' England' ' Jamaica'
 ' Guatemala' ' Portugal' ' South' ' Haiti' ' Iran' ' Thailand'
 ' Trinadad&Tobago' ' Hong' ' Yugoslavia' ' Greece' ' Honduras'
 ' Cambodia' ' Ecuador' ' Hungary' ' Scotland' ' Laos'
 ' Outlying-US(Guam-USVI-etc)']
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
class : [' <=50K' ' >50K' ' <=50K.' ' >50K.']
-----------------------------------------------------------------------------
```

- There is extra space in column name as well as in data
- There is '?' as impurity present in the data

Replace '?' with blank in the class feature

[12]:
```
df['class'] = df['class'].apply(lambda x: x.replace('.',''))
```

Remove extra space from the column name

[13]:
```
df.columns = df.columns.str.strip()
df.columns
```

[13]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'class'],
      dtype='object')
```

Remove extra space from the data

[14]:
```
df = df.applymap(lambda x: " ".join(x.split()) if isinstance(x, str) else x)
```

Replace '?' with most mode value

[15]:
```
for impure_col in ["workclass", "native-country", "occupation"]:
 frequent_value = df[impure_col].mode()[0]
 df[impure_col] = df[impure_col].replace(['?'], frequent_value)
```

Check whether '?' is present or not in the dataset

[16]:
```
df[(df['workclass'] == '?') | (df['native-country'] == '?') | (df['occupation']␣
 ↪== '?')].sum()
```

[16]:
```
age               0.0
workclass         0.0
fnlwgt            0.0
education         0.0
education-num     0.0
marital-status    0.0
occupation        0.0
relationship      0.0
race              0.0
sex               0.0
capital-gain      0.0
capital-loss      0.0
hours-per-week    0.0
native-country    0.0
class             0.0
dtype: float64
```

Check null values in the dataset

[17]:
```
df.isnull().sum()
```

[17]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
class             0
dtype: int64
```

Check duplicate values in the dataset

[18]:
```
df.duplicated().sum()
```

[18]:
```
2
```

Drop duplicates values from the dataset

[19]:
```
df.drop_duplicates(inplace=True)
```

Check duplicates after the deletion

[20]:
```
df.duplicated().sum()
```

[20]: 0

Categorical Features

[21]:
```
categorical_features = [feature for feature in df.columns if df[feature].dtypes␣
↪== 'O']
categorical_features
```

[21]:
```
['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native-country',
 'class']
```

Numerical Features

[22]:
```
numerical_features = [feature for feature in df.columns if df[feature].dtypes !
↪='O']
numerical_features
```

[22]:
```
['age',
 'fnlwgt',
 'education-num',
 'capital-gain',
 'capital-loss',
 'hours-per-week']
```

0.2 Handling of Categorical Features

[23]:
```
df[categorical_features].nunique()
```

[23]:
```
workclass          8
education         16
marital-status     7
occupation        14
relationship       6
race               5
sex                2
native-country    40
class              2
```

```
dtype: int64
```

Check unique values in each category

[24]:
```
for i in categorical_features:
  print(f"{i} : {df[i].unique()}")
```

```
workclass : ['Private' 'Federal-gov' 'Self-emp-not-inc' 'State-gov' 'Local-gov'
 'Self-emp-inc' 'Never-worked' 'Without-pay']
education : ['HS-grad' 'Some-college' '10th' '11th' 'Masters' '12th' 'Bachelors'
 'Assoc-voc' 'Doctorate' 'Prof-school' 'Assoc-acdm' '7th-8th' '9th'
 '1st-4th' '5th-6th' 'Preschool']
marital-status : ['Separated' 'Never-married' 'Married-civ-spouse' 'Married-
spouse-absent'
 'Divorced' 'Widowed' 'Married-AF-spouse']
occupation : ['Craft-repair' 'Adm-clerical' 'Prof-specialty' 'Other-service'
 'Exec-managerial' 'Handlers-cleaners' 'Tech-support' 'Transport-moving'
 'Machine-op-inspct' 'Sales' 'Farming-fishing' 'Protective-serv'
 'Priv-house-serv' 'Armed-Forces']
relationship : ['Unmarried' 'Not-in-family' 'Husband' 'Wife' 'Own-child' 'Other-
relative']
race : ['White' 'Black' 'Other' 'Asian-Pac-Islander' 'Amer-Indian-Eskimo']
sex : ['Female' 'Male']
native-country : ['Columbia' 'United-States' 'Mexico' 'China' 'Nicaragua'
'Philippines'
 'Dominican-Republic' 'Germany' 'Japan' 'Cuba' 'Poland' 'Canada' 'Vietnam'
 'Puerto-Rico' 'Ireland' 'France' 'Italy' 'Taiwan' 'El-Salvador' 'India'
 'Peru' 'England' 'Jamaica' 'Guatemala' 'Portugal' 'South' 'Haiti' 'Iran'
 'Thailand' 'Trinadad&Tobago' 'Hong' 'Yugoslavia' 'Greece' 'Honduras'
 'Cambodia' 'Ecuador' 'Hungary' 'Scotland' 'Laos'
 'Outlying-US(Guam-USVI-etc)']
class : ['<=50K' '>50K']
```

Reduce number of catgeory in marital-status

[25]:
```
df['marital-status'] = df['marital-status'].map({'Never-married' : 'Single',␣
↪'Married-civ-spouse' : 'Married',
                  'Married-spouse-absent' : 'Married','Married-AF-spouse' :␣
↪'Married', 'Divorced' : 'Divorced',
                  'Separated' : 'Separated', 'Widowed' : 'Widowed'})
```

Reduce number of catgeory in workclass

[26]:
```
df['workclass'] = df['workclass'].map({'State-gov' : 'Government',␣
↪'Self-emp-not-inc' : 'Self_Employed',
             'Private' : 'Private', 'Federal-gov' : 'Government', 'Local-gov' :␣
↪'Government',
```

```
            'Self-emp-inc' : 'Self_Employed', 'Without-pay' : 'Not_Working',␣
↪'Never-worked' : 'Not_Working'})
```

[27]:
```
df['sex'].unique()
```

[27]:
```
array(['Female', 'Male'], dtype=object)
```

Map Male to 1 and Female to 0

[28]:
```
df['sex'] = df['sex'].map({'Male' : 1, 'Female' : 0})
```

Map ">50K" to 1 and "<=50K" to 0

[29]:
```
df['class'] = df['class'].map({'>50K' : 1, '<=50K' : 0})
```

Check Correlation of numerical features

0.3 Graphical Analysis

[30]:
```
df_numerical_features = df[numerical_features]
```

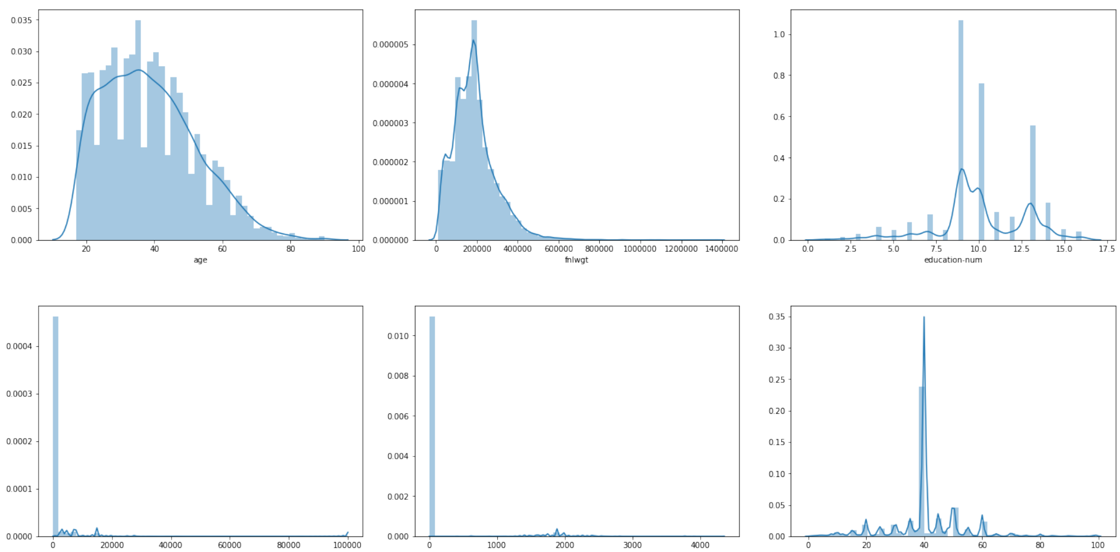
0.3.1 Numerical Features Analysis

Distplot

[31]:
```
fig, ax = plt.subplots(ncols=3, nrows=2, figsize=(20,10))
index = 0
ax = ax.flatten()
for col, value in df_numerical_features.items():
  sns.distplot(value, ax=ax[index])
  index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```

```
C:\Users\subhash\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will
be interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
C:\Users\subhash\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\subhash\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
```

```
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\subhash\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\subhash\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\subhash\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
C:\Users\subhash\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462:
UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



0.3.2 Categorical Features Analysis

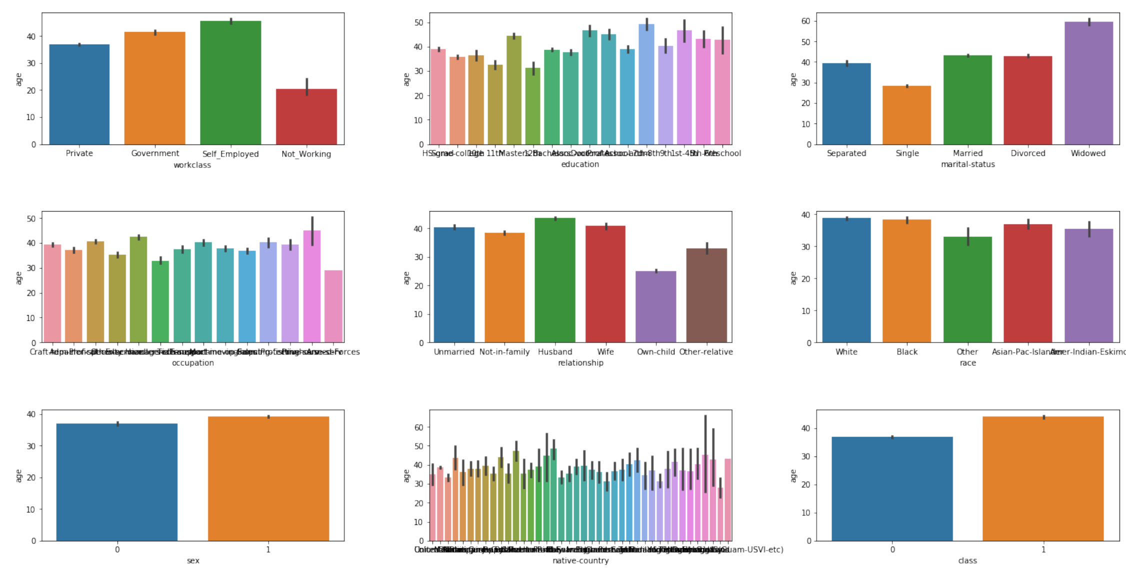
[32]:
```
df_categorical_features = df[categorical_features]
```

Barplot

[33]:
```
fig, ax = plt.subplots(ncols=3, nrows=3, figsize=(20,10))
index = 0
ax = ax.flatten()
for col, value in df_categorical_features.items():
```

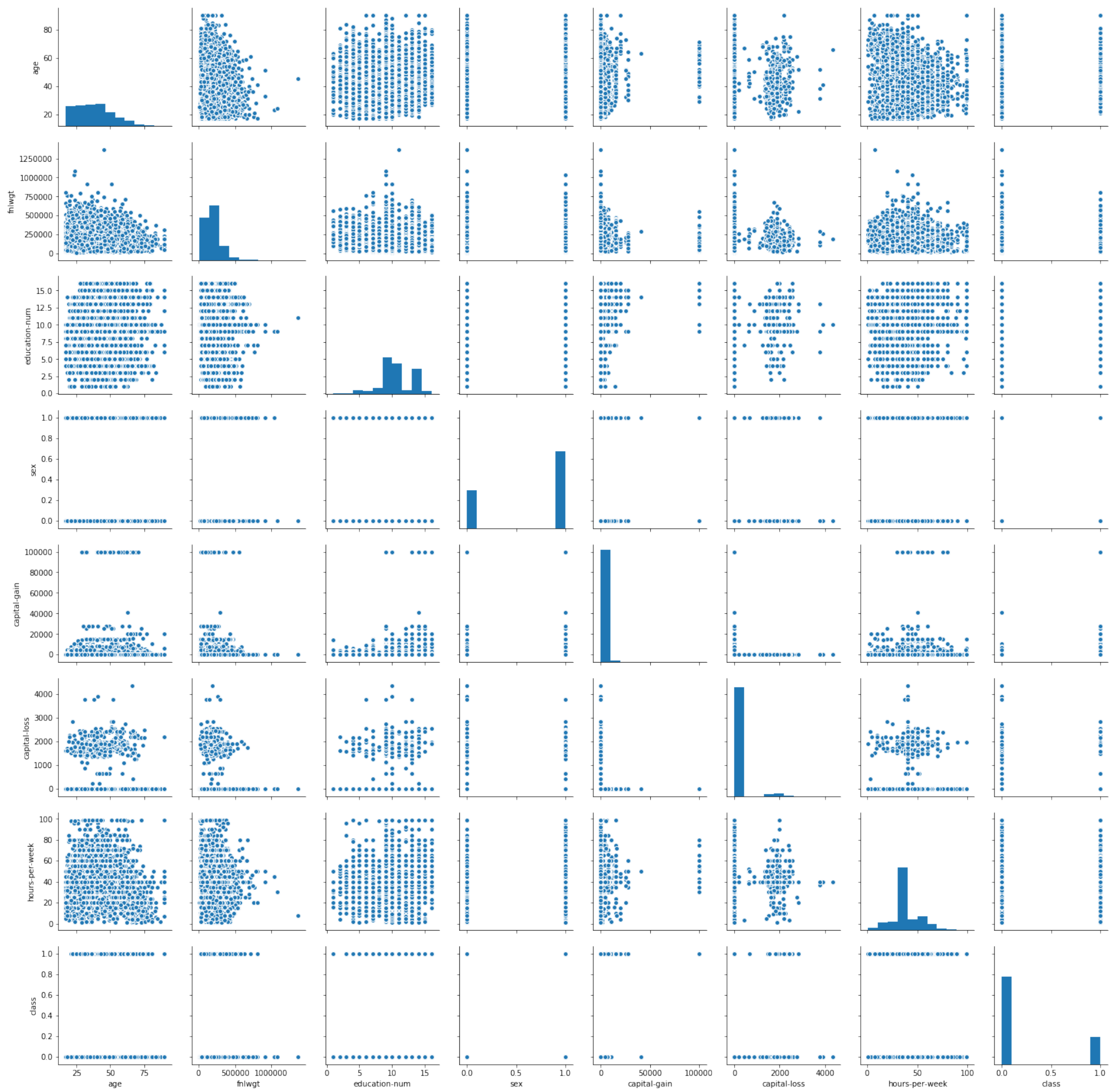
```
    sns.barplot(y = df['age'], x = df[col], data = df, ax=ax[index])
    index += 1
plt.tight_layout(pad=0.5, w_pad=0.7, h_pad=5.0)
```

```
C:\Users\subhash\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is
deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will
be interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
[34]: sns.pairplot(df)
```

```
[34]: <seaborn.axisgrid.PairGrid at 0x205edf107f0>
```



0.4 Statistical Analysis

[35]:
```
df.describe().T
```

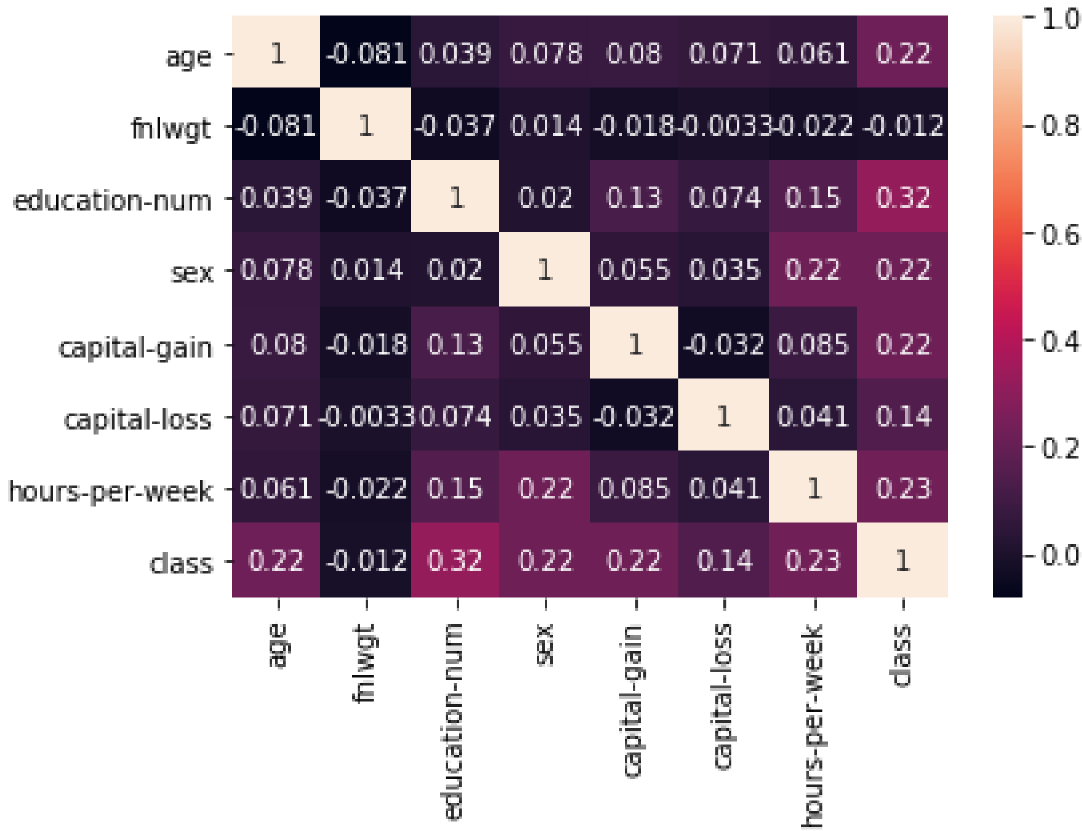
[35]:

	count	mean	std	min	25%	\
age	9998.0	38.543609	13.611437	17.0	28.0	
fnlwgt	9998.0	188353.235347	105032.757977	12285.0	116792.5	
education-num	9998.0	10.099120	2.560048	1.0	9.0	
sex	9998.0	0.670534	0.470043	0.0	0.0	
capital-gain	9998.0	970.772254	6752.734625	0.0	0.0	
capital-loss	9998.0	90.629226	411.279024	0.0	0.0	
hours-per-week	9998.0	40.414483	12.473665	1.0	40.0	

```
class          9998.0       0.235347       0.424237      0.0        0.0

                   50%        75%        max
age               37.0      47.00       90.0
fnlwgt        177933.0  236805.75  1366120.0
education-num     10.0      12.00       16.0
sex                1.0       1.00        1.0
capital-gain       0.0       0.00    99999.0
capital-loss       0.0       0.00     4356.0
hours-per-week    40.0      45.00       99.0
class              0.0       0.00        1.0
```

[36]:
```python
sns.heatmap(data = df.corr(), annot = True)
```

[36]: <matplotlib.axes._subplots.AxesSubplot at 0x205f2bc5f98>



0.5 Encoding

Frequency Encoding

[37]:
```
df.nunique()
```

[37]:
```
age                  72
workclass             4
fnlwgt             8548
education            16
education-num        16
marital-status        5
occupation           14
relationship          6
race                  5
sex                   2
capital-gain         97
capital-loss         73
hours-per-week       89
native-country       40
class                 2
dtype: int64
```

[38]:
```
for col in ['workclass', 'marital-status', 'occupation', 'relationship',␣
↪'race', 'native-country']:
  # df['workclass'] = df['workclass'].map(df.groupby("workclass").size()/
↪len(df)).round(2)
  df[col] = df[col].map(df.groupby(col).size()/len(df)).round(2)
```

Drop "education" column because we have one more columns as "eduction-num" which is encoded to "eductaion" column

[39]:
```
df.drop('education', axis = 1, inplace = True)
```

[40]:
```
# X = df.iloc[ : , :-1]
# y = df.iloc[ : , -1]
```

[41]:
```
# X.shape
```

[42]:
```
# y.shape
```

0.6 Save Preprocess Model Data Using Pickle

[43]:
```
# preprocess_model = [X_train,y_train,X_test,y_test]
preprocess_model = [df]
```

[44]:
```
import pickle
```

[45]:
```
pickle.dump(preprocess_model,␣
↪open('Census_Income_Classification_Preprocess_Model.pkl','wb'))
```

[46]:
```
preprocess_model = pickle.
→load(open('Census_Income_Classification_Preprocess_Model.pkl','rb'))
```

Note * We have successfully stored our scaled data into pickel file so we can use it further in other file by just importing it

0.7 Save Data into MongoDb

[47]:
```
# !pip install pymongo
```

[48]:
```
# import pymongo
# from pymongo import MongoClient
```

[49]:
```
# client = pymongo.MongoClient("mongodb+srv://subhashdixit17:Anushka27@cluster0.
→elq8eyt.mongodb.net/?retryWrites=true&w=majority")
```

[50]:
```
# db=client['Census_Income_Preprocessed_Data']
# collections = db['Training__Independent_and_Dependent_Dataset']
```

[51]:
```
# data_json = df.to_dict('records')
# collections.insert_many(data_json)
```

0.8 Load Preprocessed data using MongoDb

[52]:
```
# Getting all records from mongodb
# imported_data = collections.find()
# imported_data = pd.DataFrame(imported_data)
```

0.9 Dropping Unnecessary features

[53]:
```
# data = imported_data.drop(['_id'], axis=1)
```

0.10 Spliting Independent and Dependent Features

[54]:
```
# X = data.iloc[:, 0:13]
# y= data.iloc[:, -1]
X = df.iloc[:, 0:13]
y= df.iloc[:, -1]
```

0.11 Train Test Split

[55]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=7,test_size=0.
↪33)
```

0.12 Scaling

[56]:
```
from sklearn.preprocessing import StandardScaler
```

[57]:
```
scaler=StandardScaler()
```

[58]:
```
X_train = scaler.fit_transform(X_train)
```

[59]:
```
X_test = scaler.transform(X_test)
```

0.13 VIF Check

- **To check multicollinearity**

[60]:
```
# X_train = pd.DataFrame(X_train)
```

[61]:
```
# from statsmodels.stats.outliers_influence import variance_inflation_factor
# vif = [variance_inflation_factor(X_train.values, i) for i in range(X_train.
↪shape[1])]
# print(X_train.columns)
# print(vif)
```

[62]:
```
# while (max(vif) > 5):
#     indx = vif.index(max(vif)) #Get the index of variable with highest VIF
#     print(indx)
#     X_train.drop(X_train.columns[indx],axis = 1, inplace = True)
#     vif = [variance_inflation_factor(X_train.values, i) for i in␣
↪range(X_train.shape[1])]
# vif = [variance_inflation_factor(X_train.values, i) for i in range(X_train.
↪shape[1])]
# print(X_train.columns)
# print(vif)
```

[63]:
```
# X_train.head()
```

[64]:
```
# X_test = X_test[X_train.columns]
```

[65]:
```
# X_test.head()
```

1 Model Creation

1.1 GridSearchCV For SVC

[66]:
```
from sklearn.model_selection import GridSearchCV
```

[67]:
```
# param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel':
↪['linear','rbf', 'poly', 'sigmoid']}
```

[68]:
```
# model_GRID_SVR = GridSearchCV(SVC(),param_grid,refit=True,verbose=3)
# model_GRID_SVR.fit(X_train,y_train)
```

[69]:
```
# print(model_GRID_SVR.best_estimator_)
```

[70]:
```
# GRID_SVR_train_score = model_GRID_SVR.score(X_train, y_train)
# GRID_SVR_train_score
```

2 All Model Creation

[71]:
```
# Model Mapping
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
param_grid = {'kernel':kernels}
## We will train that models
models = {
 1: LogisticRegression(),
#  2: LinearSVC()
 2: SVC(kernel=kernels[0]),
 3: SVC(kernel=kernels[1]),
 4: SVC(kernel=kernels[2]),
 5: SVC(kernel=kernels[3]),
 6: GridSearchCV(estimator=SVC(), param_grid=param_grid, n_jobs=-1) # HyperParam
}
```

[72]:
```
map_keys = list(models.keys())
```

[73]:
```
# Get model name using id from linear_model_collection
def get_model_building_technique_name(num):
 if num == 1:
  return 'Logistic Regression()'
 if num == 2:
  # return 'LinearSVC()'
  return "SVC(kernel='linear')"
 if num == 3:
  return "SVC(kernel='poly', cache_size=7000)"
 if num == 4:
```

```
    return "SVC(kernel='rbf', cache_size=7000)"
  if num == 5:
    return "SVC(kernel='sigmoid', cache_size=7000)"
  if num == 6:
    return 'GridSearchCV Estimator SVC'
  return ''
```

[74]:
```
results = [];
for key_index in range(len(map_keys)):
  key = map_keys[key_index]
  model = models[key]
  print(key_index)
  model.fit(X_train, y_train)

  '''Test Accuracy'''
  y_pred = model.predict(X_test)

  Accuracy_Test = accuracy_score(y_test, y_pred)
  conf_mat_Test = confusion_matrix(y_test, y_pred)
  true_positive_Test = conf_mat_Test[0][0]
  false_positive_Test = conf_mat_Test[0][1]
  false_negative_Test = conf_mat_Test[1][0]
  true_negative__Test = conf_mat_Test[1][1]
  Precision_Test = true_positive_Test /(true_positive_Test +␣
↪false_positive_Test)
  Recall_Test = true_positive_Test/(true_positive_Test + false_negative_Test)
  F1_Score_Test = 2*(Recall_Test * Precision_Test) / (Recall_Test +␣
↪Precision_Test)
  AUC_Test = roc_auc_score(y_test, y_pred)

  '''Train Accuracy'''
  y_pred_train = model.predict(X_train)

  Accuracy_Train = accuracy_score(y_train, y_pred_train)
  conf_mat_Train = confusion_matrix(y_train, y_pred_train)
  true_positive_Train = conf_mat_Train[0][0]
  false_positive_Train = conf_mat_Train[0][1]
  false_negative_Train = conf_mat_Train[1][0]
  true_negative__Train = conf_mat_Train[1][1]
  Precision_Train = true_positive_Train /(true_positive_Train +␣
↪false_positive_Train)
  Recall_Train = true_positive_Train/(true_positive_Train +␣
↪false_negative_Train)
  F1_Score_Train = 2*(Recall_Train * Precision_Train) / (Recall_Train +␣
↪Precision_Train)
  AUC_Train = roc_auc_score(y_train, y_pred_train)
```

```
    results.append({
        'Model Name' : get_model_building_technique_name(key),
        'Trained Model' : model,
        'Accuracy_Test' : Accuracy_Test,
        'Precision_Test' : Precision_Test,
        'Recall_Test' : Recall_Test,
        'F1_Score_Test' : F1_Score_Test,
        'AUC_Test' : AUC_Test,
        'Accuracy_Train' : Accuracy_Train,
        'Precision_Train' : Precision_Train,
        'Recall_Train' : Recall_Train,
        'F1_Score_Train' : F1_Score_Train,
        'AUC_Train' : AUC_Train
        })
```

```
0
1
2
3
4
5
```

[75]:

```
result_df = pd.DataFrame(results)
result_df
```

[75]:

```
   AUC_Test  AUC_Train  Accuracy_Test  Accuracy_Train  F1_Score_Test  \
0  0.736090   0.728649       0.844848        0.838161       0.903030
1  0.716071   0.710988       0.840606        0.837414       0.901498
2  0.723047   0.741667       0.842727        0.856674       0.902499
3  0.739576   0.756058       0.848788        0.859958       0.905653
4  0.694186   0.682265       0.780909        0.772768       0.857425
5  0.739576   0.756058       0.848788        0.859958       0.905653

   F1_Score_Train                             Model Name  Precision_Test  \
0        0.898235                   Logistic Regression()        0.936371
1        0.899232                    SVC(kernel='linear')        0.945405
2        0.910847     SVC(kernel='poly', cache_size=7000)        0.943441
3        0.912139      SVC(kernel='rbf', cache_size=7000)        0.940691
4        0.851454  SVC(kernel='sigmoid', cache_size=7000)        0.853888
5        0.912139                GridSearchCV Estimator SVC        0.940691

   Precision_Train  Recall_Test  Recall_Train  \
0         0.938223     0.871982      0.861516
1         0.952932     0.861489      0.851261
2         0.961757     0.864962      0.865056
3         0.954893     0.873132      0.873050
4         0.855462     0.860990      0.847484
```

```
5         0.954893     0.873132      0.873050

                                              Trained Model
0  LogisticRegression(C=1.0, class_weight=None, d…
1  SVC(C=1.0, cache_size=200, class_weight=None, …
2  SVC(C=1.0, cache_size=200, class_weight=None, …
3  SVC(C=1.0, cache_size=200, class_weight=None, …
4  SVC(C=1.0, cache_size=200, class_weight=None, …
5  GridSearchCV(cv=None, error_score='raise',\n  …
```

2.1 Test Accuracy

[76]:
```
result_df_test = result_df.iloc[: , [6,11,2,4,7,9,0]]
result_df_test
```

[76]:
```
                               Model Name  \
0                    Logistic Regression()
1                     SVC(kernel='linear')
2      SVC(kernel='poly', cache_size=7000)
3       SVC(kernel='rbf', cache_size=7000)
4  SVC(kernel='sigmoid', cache_size=7000)
5                 GridSearchCV Estimator SVC

                                              Trained Model  Accuracy_Test  \
0  LogisticRegression(C=1.0, class_weight=None, d…       0.844848
1  SVC(C=1.0, cache_size=200, class_weight=None, …       0.840606
2  SVC(C=1.0, cache_size=200, class_weight=None, …       0.842727
3  SVC(C=1.0, cache_size=200, class_weight=None, …       0.848788
4  SVC(C=1.0, cache_size=200, class_weight=None, …       0.780909
5  GridSearchCV(cv=None, error_score='raise',\n  …       0.848788

   F1_Score_Test  Precision_Test  Recall_Test  AUC_Test
0       0.903030        0.936371     0.871982  0.736090
1       0.901498        0.945405     0.861489  0.716071
2       0.902499        0.943441     0.864962  0.723047
3       0.905653        0.940691     0.873132  0.739576
4       0.857425        0.853888     0.860990  0.694186
5       0.905653        0.940691     0.873132  0.739576
```

2.2 Train Accuracy

[77]:
```
result_df_train = result_df.iloc[: , [6,11,3,5,9,10,1]]
result_df_train
```

[77]:

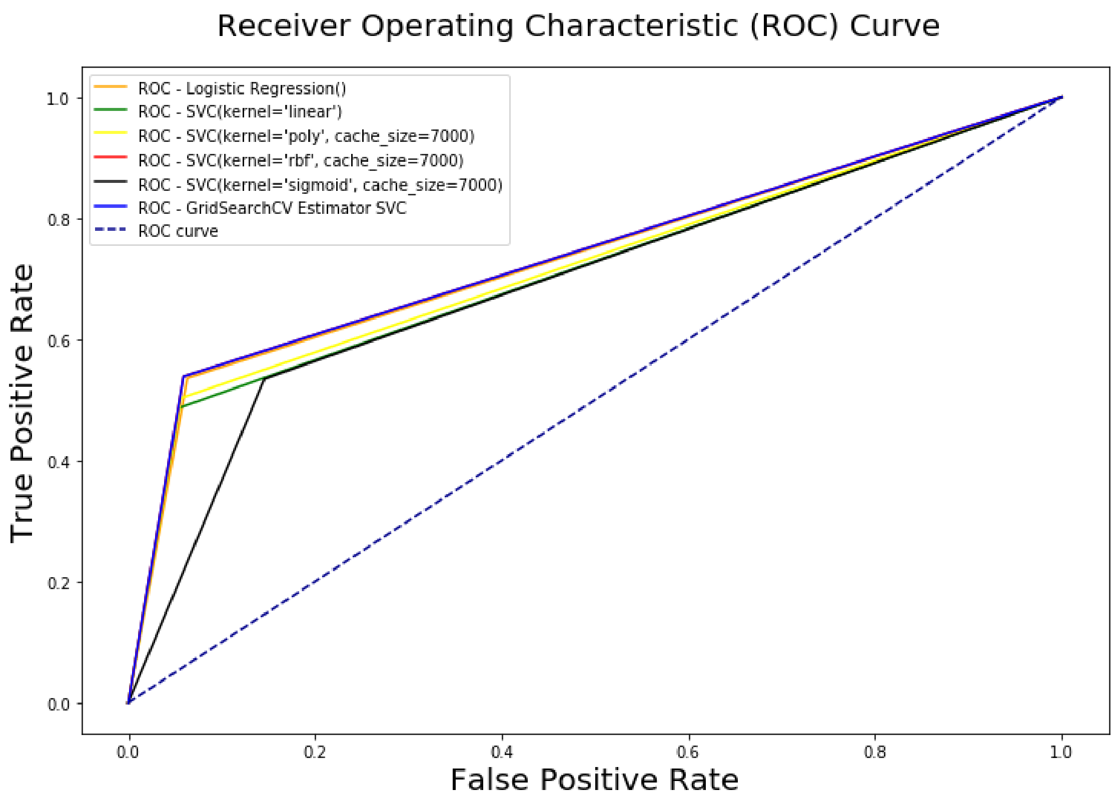
```
                             Model Name  \
0                  Logistic Regression()
1                    SVC(kernel='linear')
2     SVC(kernel='poly', cache_size=7000)
3      SVC(kernel='rbf', cache_size=7000)
4  SVC(kernel='sigmoid', cache_size=7000)
5              GridSearchCV Estimator SVC

                                    Trained Model  Accuracy_Train  \
0  LogisticRegression(C=1.0, class_weight=None, d…        0.838161
1  SVC(C=1.0, cache_size=200, class_weight=None, …        0.837414
2  SVC(C=1.0, cache_size=200, class_weight=None, …        0.856674
3  SVC(C=1.0, cache_size=200, class_weight=None, …        0.859958
4  SVC(C=1.0, cache_size=200, class_weight=None, …        0.772768
5  GridSearchCV(cv=None, error_score='raise',\n  …        0.859958

   F1_Score_Train  Recall_Test  Recall_Train  AUC_Train
0        0.898235     0.871982      0.861516   0.728649
1        0.899232     0.861489      0.851261   0.710988
2        0.910847     0.864962      0.865056   0.741667
3        0.912139     0.873132      0.873050   0.756058
4        0.851454     0.860990      0.847484   0.682265
5        0.912139     0.873132      0.873050   0.756058
```

2.3 ROC Curve for all the Model

[78]:

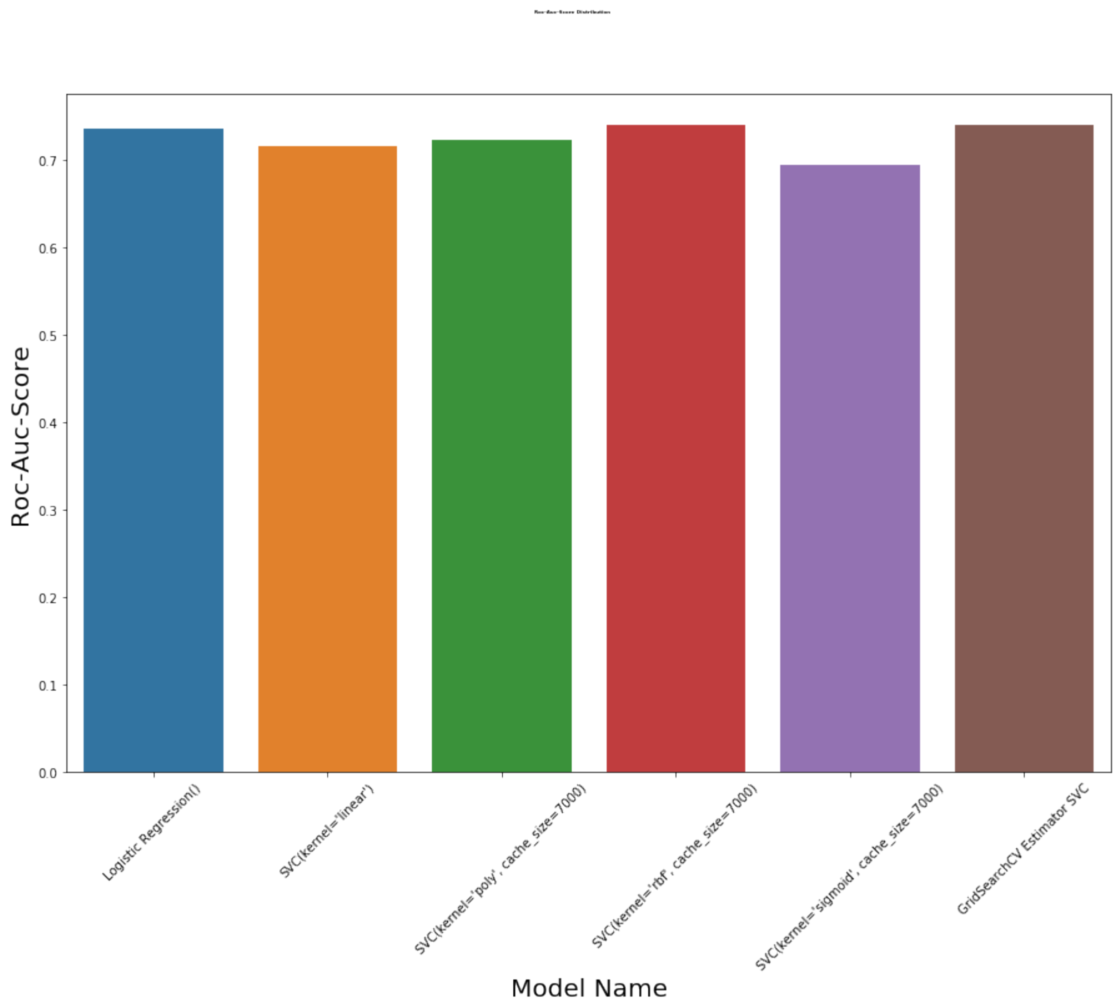
```python
fpr_dict = {}
tpr_dict = {}
for i in range(6):
    model_pred = result_df['Trained Model'][i].predict(X_test)
    fpr, tpr, thresholds = roc_curve(y_test, model_pred)
    fpr_dict[i] = fpr
    tpr_dict[i] = tpr

plt.figure(figsize=(12,8))
plt.suptitle('\nReceiver Operating Characteristic (ROC) Curve', fontsize=20)
plt.plot(fpr_dict[0], tpr_dict[0], color='orange', label=f"ROC -␣
 ↪{result_df['Model Name'][0]}")
plt.plot(fpr_dict[1], tpr_dict[1], color='green', label=f"ROC -␣
 ↪{result_df['Model Name'][1]}")
plt.plot(fpr_dict[2], tpr_dict[2], color='yellow', label=f"ROC -␣
 ↪{result_df['Model Name'][2]}")
plt.plot(fpr_dict[3], tpr_dict[3], color='red', label=f"ROC - {result_df['Model␣
 ↪Name'][3]}")
plt.plot(fpr_dict[4], tpr_dict[4], color='black', label=f"ROC -␣
 ↪{result_df['Model Name'][4]}")
```

```
plt.plot(fpr_dict[5], tpr_dict[5], color='blue', label=f"ROC -␣
↪{result_df['Model Name'][5]}")
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--',label='ROC curve')
plt.xlabel('False Positive Rate',fontdict={'fontsize': 20})
plt.ylabel('True Positive Rate',fontdict={'fontsize': 20})
plt.legend()
plt.show()
```



2.4 Checking Best Model

[79]:
```
plt.figure(figsize=(15,10))
plt.suptitle('\nRoc-Auc-Score Distribution\n\n', fontsize=4, fontweight='bold')
sns.barplot(data=result_df, x='Model Name', y='AUC_Test')
plt.xlabel('Model Name',fontdict={'fontsize': 20})
plt.ylabel('Roc-Auc-Score',fontdict={'fontsize': 20})
plt.xticks(rotation=45)
plt.show()
```



[80]:
```
Best_Model_Name = result_df['Trained Model'][result_df[result_df['AUC_Test'] == max(result_df['AUC_Test'])]['Trained Model'].index[0]]
Best_Model_Index = result_df['Trained Model'][result_df[result_df['AUC_Test'] == max(result_df['AUC_Test'])]['Trained Model'].index].index[0]
Best_Model_Name
```

[80]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

2.5 Save Best Model

[82]:
```
import pickle
Best_Trained_model = Best_Model_Name
with open('Census_Income_Classification.sav', 'wb') as best_model_pickle:
 pickle.dump(Best_Trained_model, best_model_pickle)
```

**

The End

**