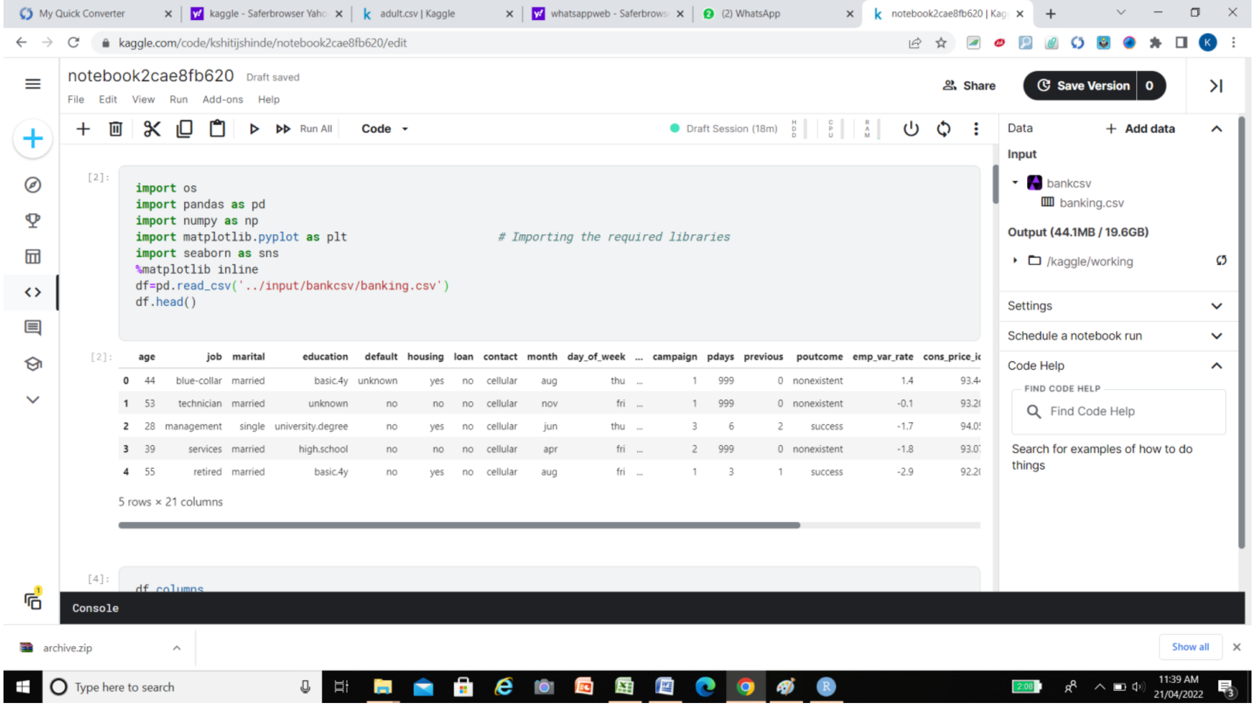
Group A
Assignment No. 5.2

Aim: Data Analytics 2 : Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given set.

OUTPUT :



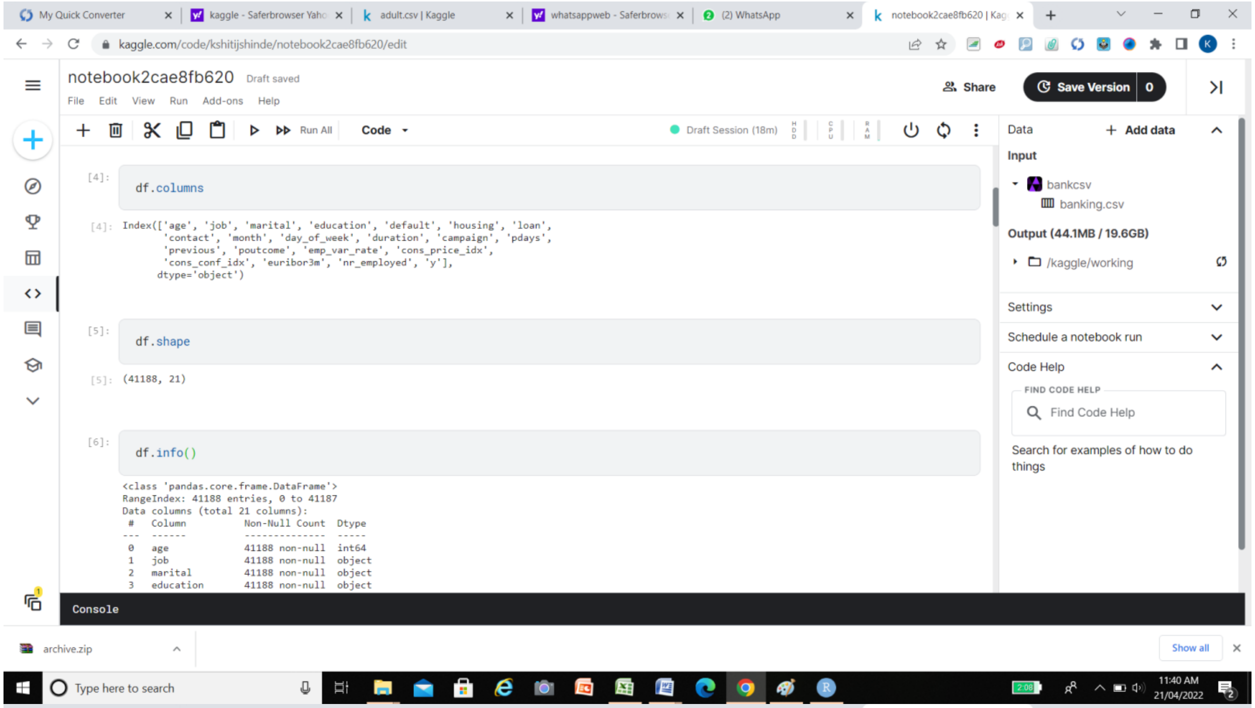
```
[2]:
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt                    # Importing the required libraries
import seaborn as sns
%matplotlib inline
df=pd.read_csv('../input/bankcsv/banking.csv')
df.head()
```

[2]:

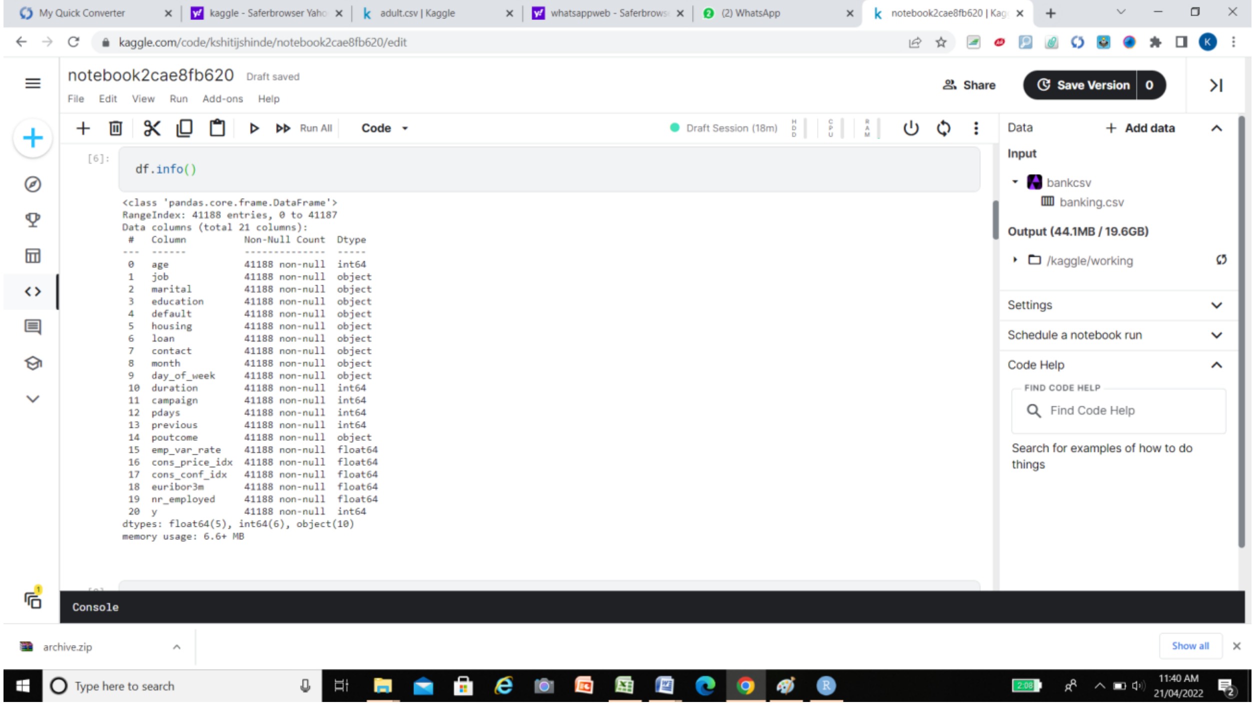
	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp_var_rate	cons_price_i
0	44	blue-collar	married	basic.4y	unknown	yes	no	cellular	aug	thu	...	1	999	0	nonexistent	1.4	93.4
1	53	technician	married	unknown	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	-0.1	93.2
2	28	management	single	university.degree	no	yes	no	cellular	jun	thu	...	3	6	2	success	-1.7	94.0
3	39	services	married	high.school	no	no	no	cellular	apr	fri	...	2	999	0	nonexistent	-1.8	93.0
4	55	retired	married	basic.4y	no	yes	no	cellular	aug	fri	...	1	3	1	success	-2.9	92.2

5 rows × 21 columns

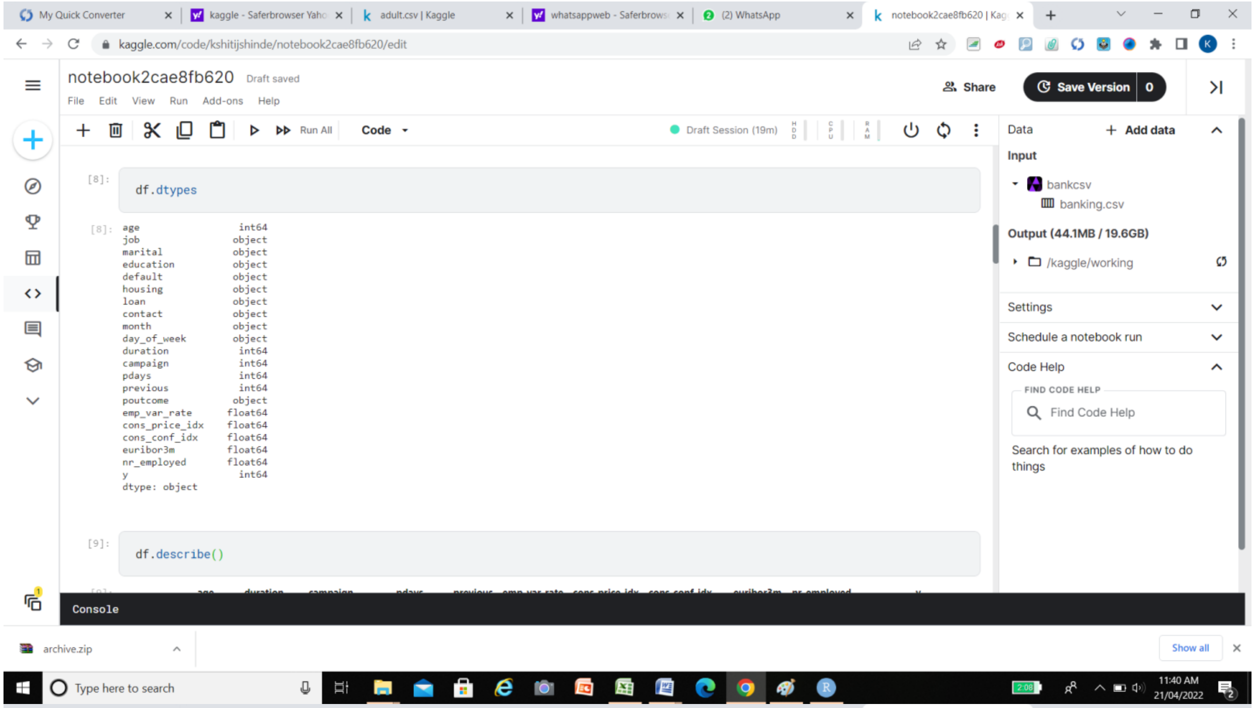
```
[4]:
df.columns
```



```
[4]:  df.columns

[4]:  Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
             'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
             'previous', 'poutcome', 'emp_var_rate', 'cons_price_idx',
             'cons_conf_idx', 'euribor3m', 'nr_employed', 'y'],
            dtype='object')

[5]:  df.shape

[5]:  (41188, 21)

[6]:  df.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 41188 entries, 0 to 41187
      Data columns (total 21 columns):
       #   Column          Non-Null Count  Dtype
      ---  ------          --------------  -----
       0   age             41188 non-null  int64
       1   job             41188 non-null  object
       2   marital         41188 non-null  object
       3   education       41188 non-null  object
```



```
[6]:  df.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 41188 entries, 0 to 41187
      Data columns (total 21 columns):
       #   Column          Non-Null Count  Dtype
      ---  ------          --------------  -----
       0   age             41188 non-null  int64
       1   job             41188 non-null  object
       2   marital         41188 non-null  object
       3   education       41188 non-null  object
       4   default         41188 non-null  object
       5   housing         41188 non-null  object
       6   loan            41188 non-null  object
       7   contact         41188 non-null  object
       8   month           41188 non-null  object
       9   day_of_week     41188 non-null  object
       10  duration        41188 non-null  int64
       11  campaign        41188 non-null  int64
       12  pdays           41188 non-null  int64
       13  previous        41188 non-null  int64
       14  poutcome        41188 non-null  object
       15  emp_var_rate    41188 non-null  float64
       16  cons_price_idx  41188 non-null  float64
       17  cons_conf_idx   41188 non-null  float64
       18  euribor3m       41188 non-null  float64
       19  nr_employed     41188 non-null  float64
       20  y               41188 non-null  int64
      dtypes: float64(5), int64(6), object(10)
      memory usage: 6.6+ MB
```



[8]:

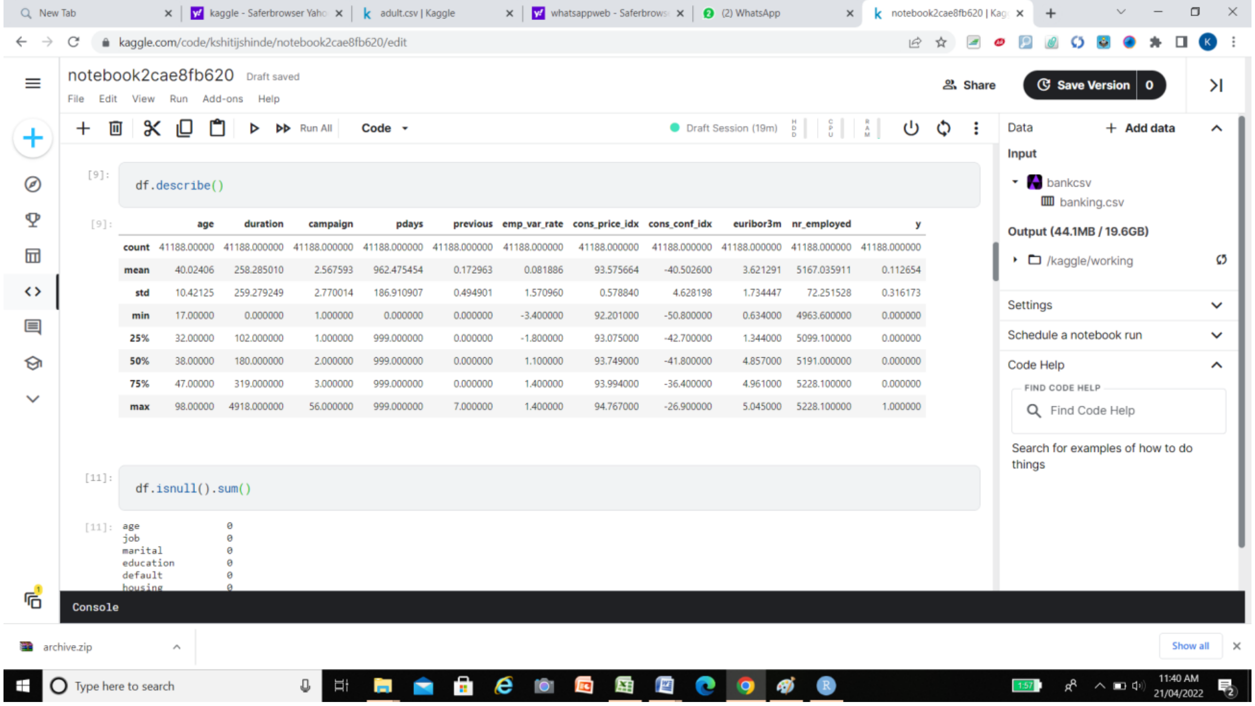
```
df.dtypes
```

[8]:

```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp_var_rate      float64
cons_price_idx    float64
cons_conf_idx     float64
euribor3m         float64
nr_employed       float64
y                   int64
dtype: object
```

[9]:

```
df.describe()
```



[9]:

```
df.describe()
```

[9]:

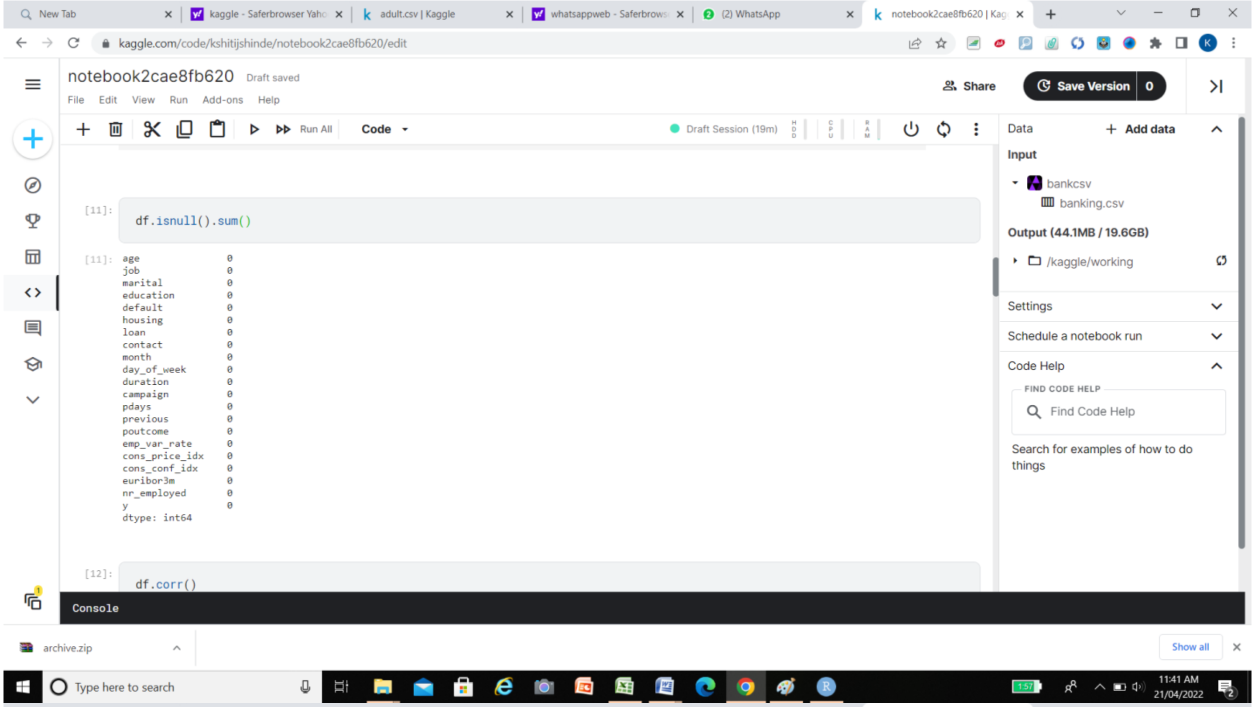
	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911	0.112654
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528	0.316173
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000	0.000000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000	0.000000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000	0.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000	0.000000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000	1.000000

[11]:

```
df.isnull().sum()
```

[11]:

```
age          0
job          0
marital      0
education    0
default      0
housing      0
```



[11]:

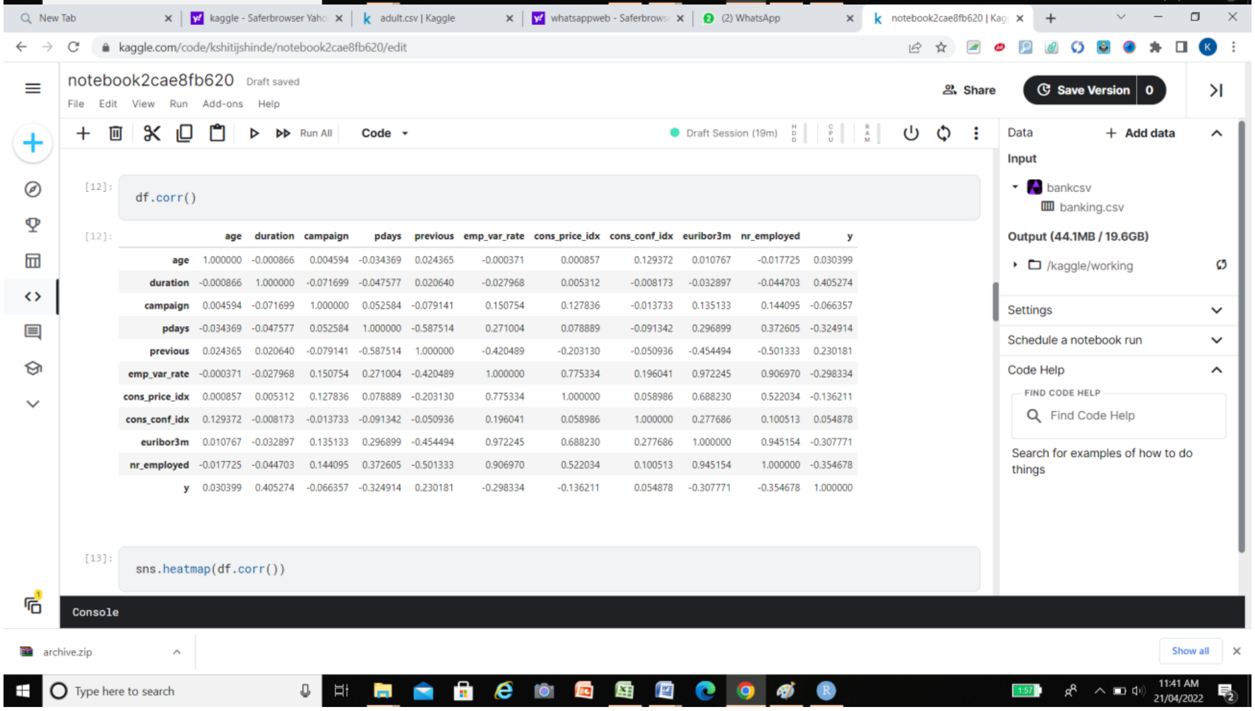
```
df.isnull().sum()
```

[11]:

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp_var_rate      0
cons_price_idx    0
cons_conf_idx     0
euribor3m         0
nr_employed       0
y                 0
dtype: int64
```

[12]:

```
df.corr()
```



[12]:

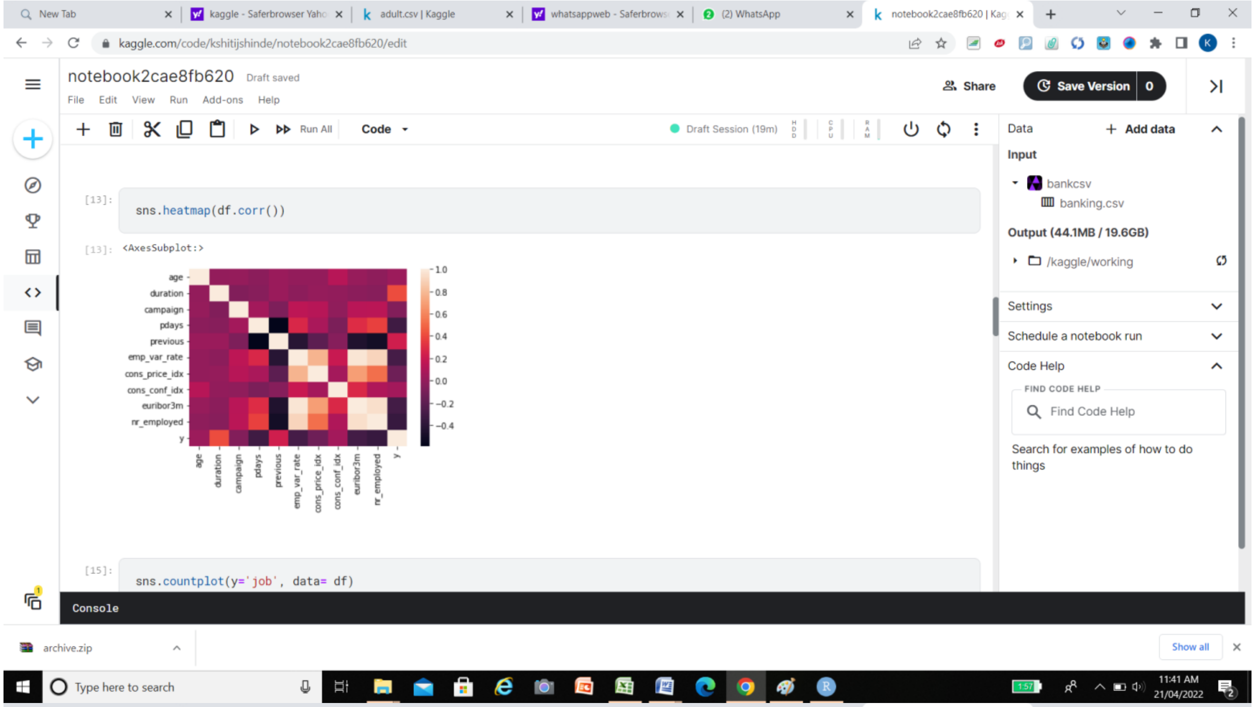
```
df.corr()
```

[12]:

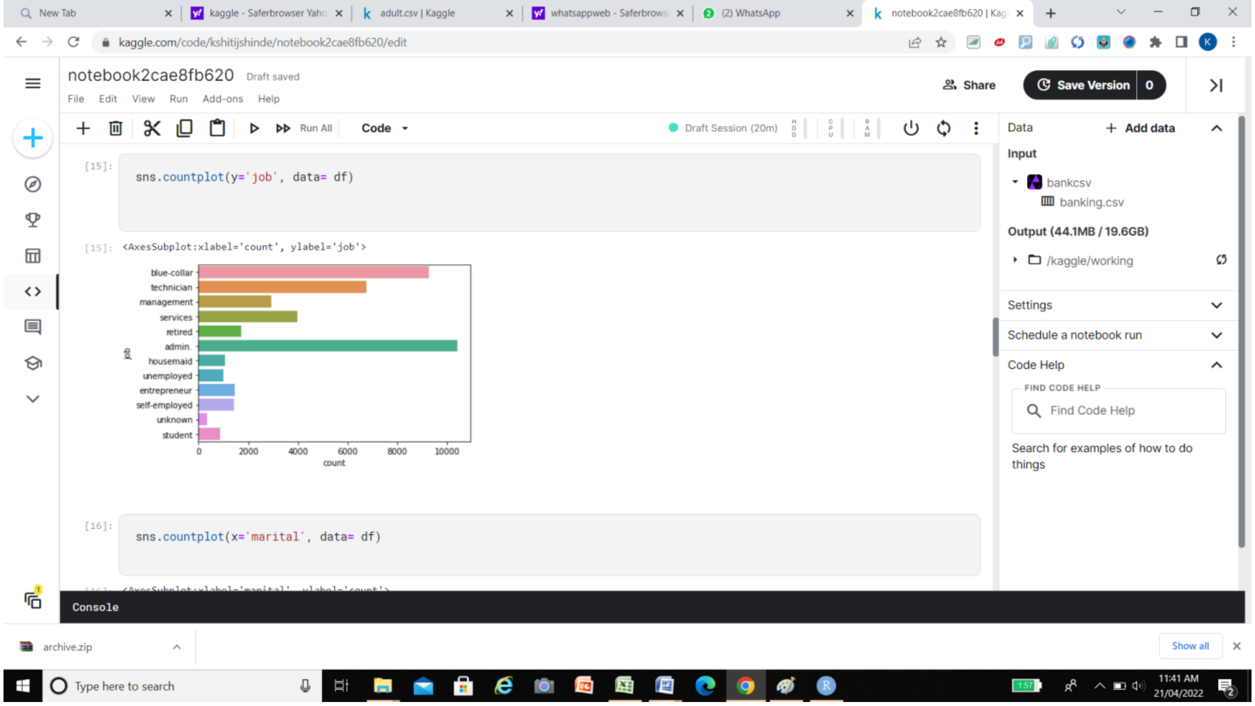
	age	duration	campaign	pdays	previous	emp_var_rate	cons_price_idx	cons_conf_idx	euribor3m	nr_employed	y
age	1.000000	-0.000866	0.004594	-0.034369	0.024365	-0.000371	0.000857	0.129372	0.010767	-0.017725	0.030399
duration	-0.000866	1.000000	-0.071699	-0.047577	0.020640	-0.027968	0.005312	-0.008173	-0.032897	-0.044703	0.405274
campaign	0.004594	-0.071699	1.000000	0.052584	-0.079141	0.150754	0.127836	-0.013733	0.135133	0.144095	-0.066357
pdays	-0.034369	-0.047577	0.052584	1.000000	-0.587514	0.271004	0.078889	-0.091342	0.296899	0.372605	-0.324914
previous	0.024365	0.020640	-0.079141	-0.587514	1.000000	-0.420489	-0.203130	-0.050936	-0.454494	-0.501333	0.230181
emp_var_rate	-0.000371	-0.027968	0.150754	0.271004	-0.420489	1.000000	0.775334	0.196041	0.972245	0.906970	-0.298334
cons_price_idx	0.000857	0.005312	0.127836	0.078889	-0.203130	0.775334	1.000000	0.058986	0.688230	0.522034	-0.136211
cons_conf_idx	0.129372	-0.008173	-0.013733	-0.091342	-0.050936	0.196041	0.058986	1.000000	0.277686	0.100513	0.054878
euribor3m	0.010767	-0.032897	0.135133	0.296899	-0.454494	0.972245	0.688230	0.277686	1.000000	0.945154	-0.307771
nr_employed	-0.017725	-0.044703	0.144095	0.372605	-0.501333	0.906970	0.522034	0.100513	0.945154	1.000000	-0.354678
y	0.030399	0.405274	-0.066357	-0.324914	0.230181	-0.298334	-0.136211	0.054878	-0.307771	-0.354678	1.000000

[13]:

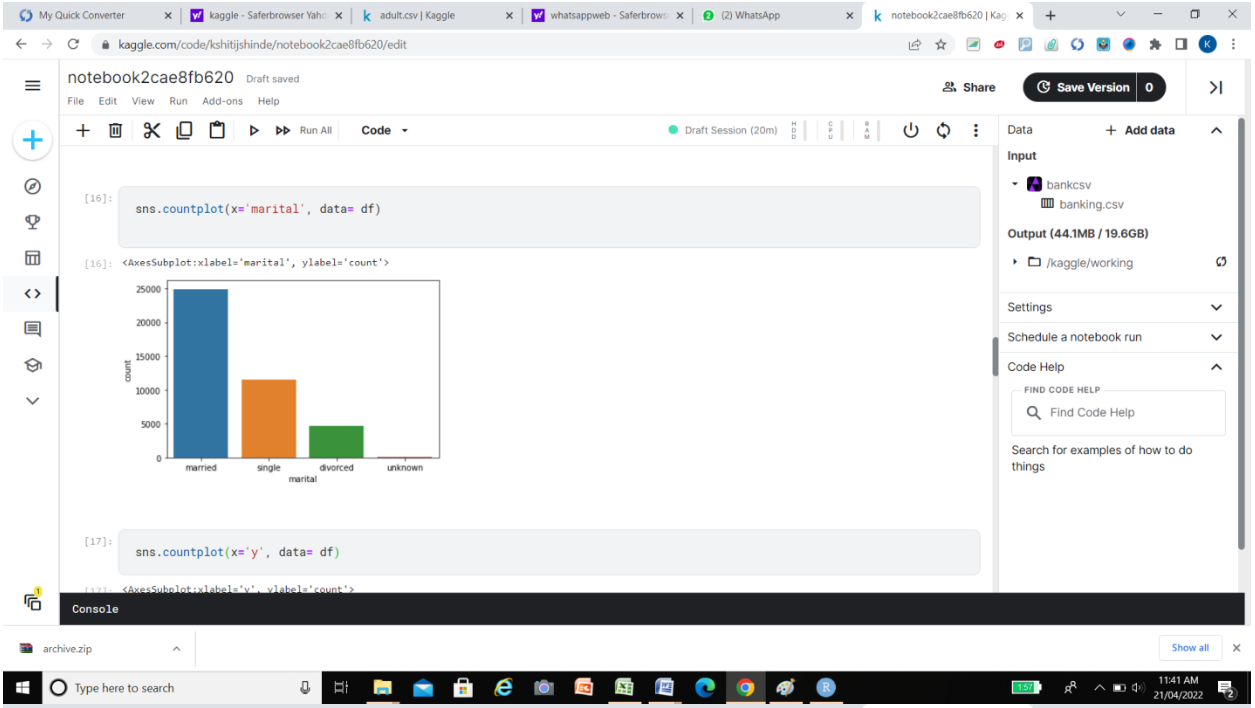
```
sns.heatmap(df.corr())
```



```
[13]: sns.heatmap(df.corr())

[13]: <AxesSubplot:>

[15]: sns.countplot(y='job', data= df)
```

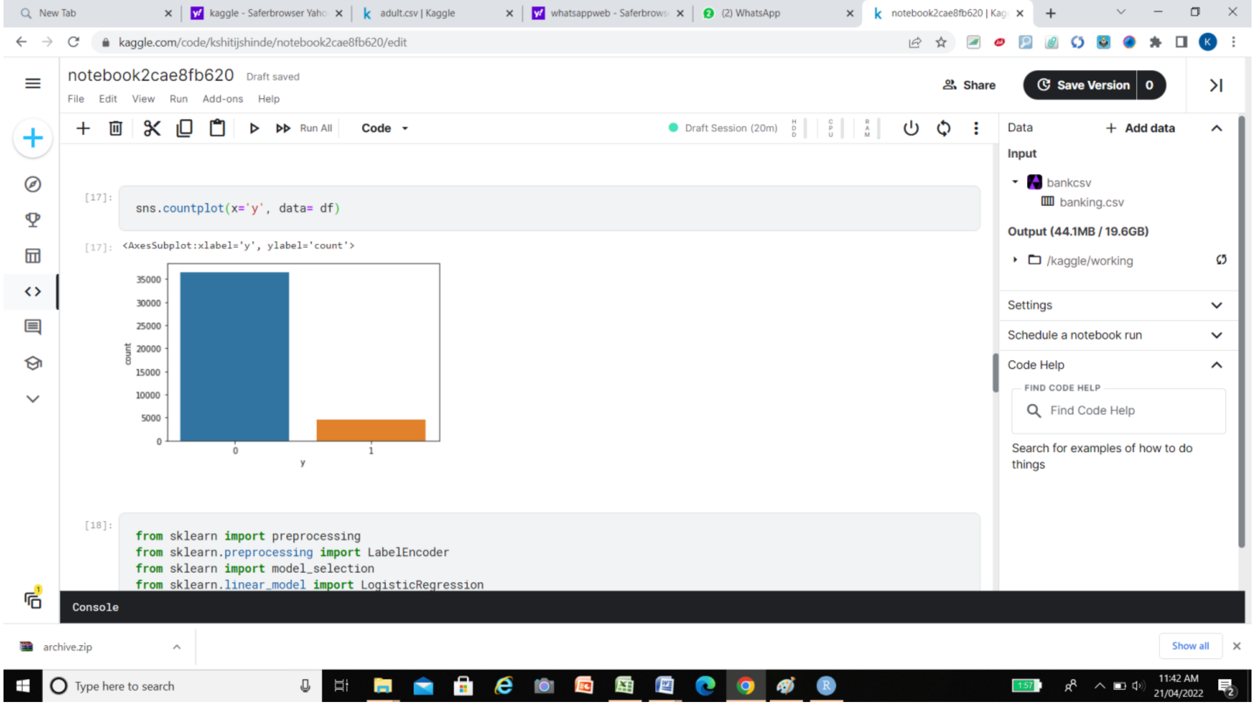


```
[15]: sns.countplot(y='job', data= df)

[15]: <AxesSubplot:xlabel='count', ylabel='job'>

[16]: sns.countplot(x='marital', data= df)

[16]: <AxesSubplot:xlabel='marital', ylabel='count'>
```



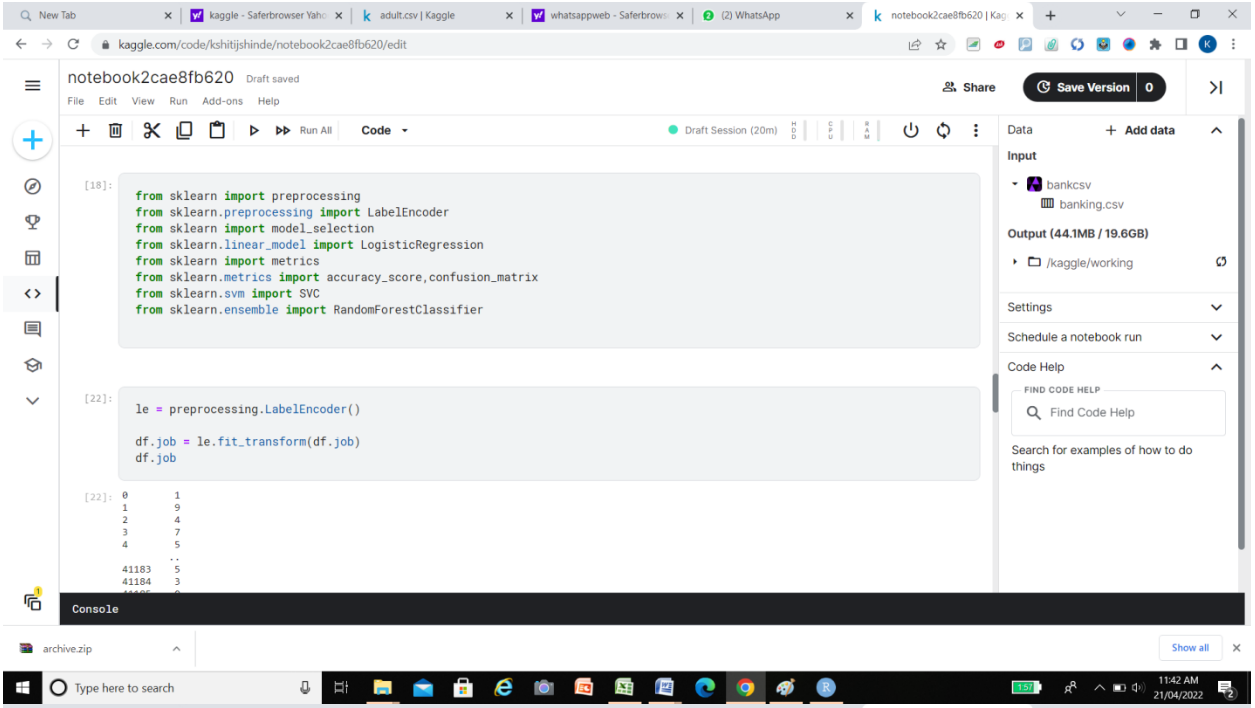
```
[16]: sns.countplot(x='marital', data= df)

[16]: <AxesSubplot:xlabel='marital', ylabel='count'>
```

```
[17]: sns.countplot(x='y', data= df)

[17]: <AxesSubplot:xlabel='y', ylabel='count'>
```



```
[17]: sns.countplot(x='y', data= df)

[17]: <AxesSubplot:xlabel='y', ylabel='count'>
```

```
[18]: from sklearn import preprocessing
      from sklearn.preprocessing import LabelEncoder
      from sklearn import model_selection
      from sklearn.linear_model import LogisticRegression
```



[18]:

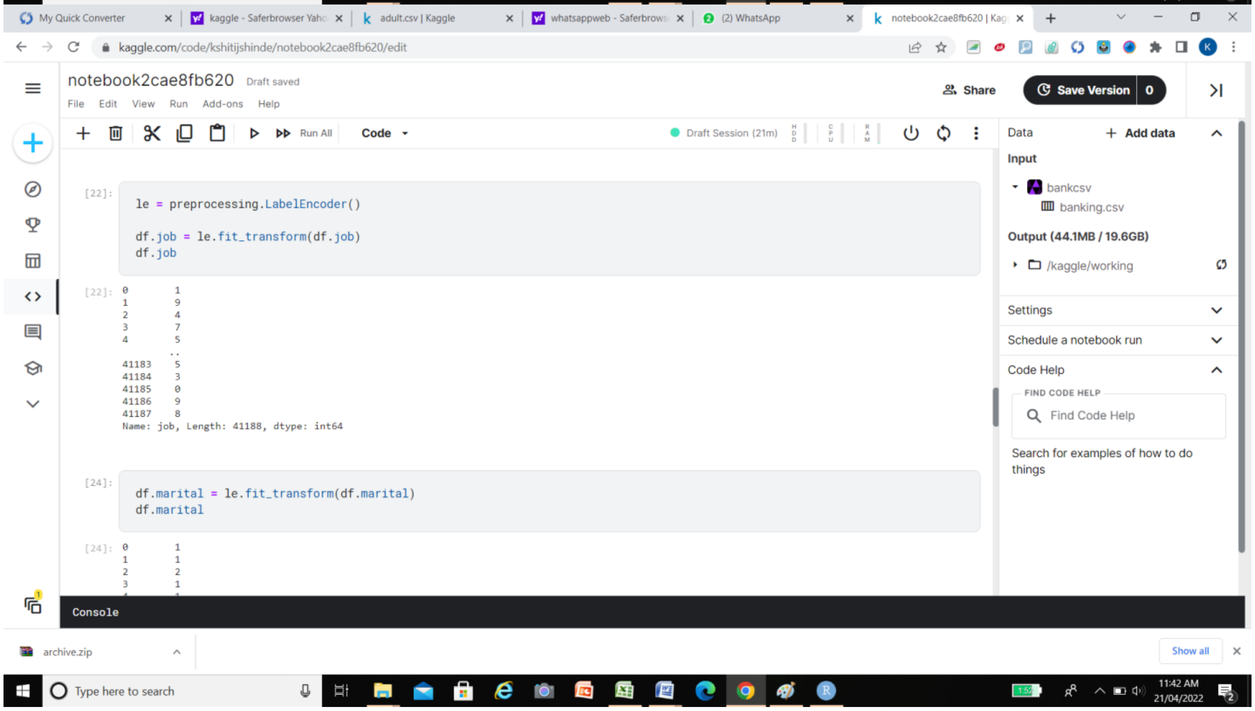
```python
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

[22]:

```python
le = preprocessing.LabelEncoder()

df.job = le.fit_transform(df.job)
df.job
```

[22]:

```
0        1
1        9
2        4
3        7
4        5
        ..
41183    5
41184    3
```



[22]:

```python
le = preprocessing.LabelEncoder()

df.job = le.fit_transform(df.job)
df.job
```

[22]:

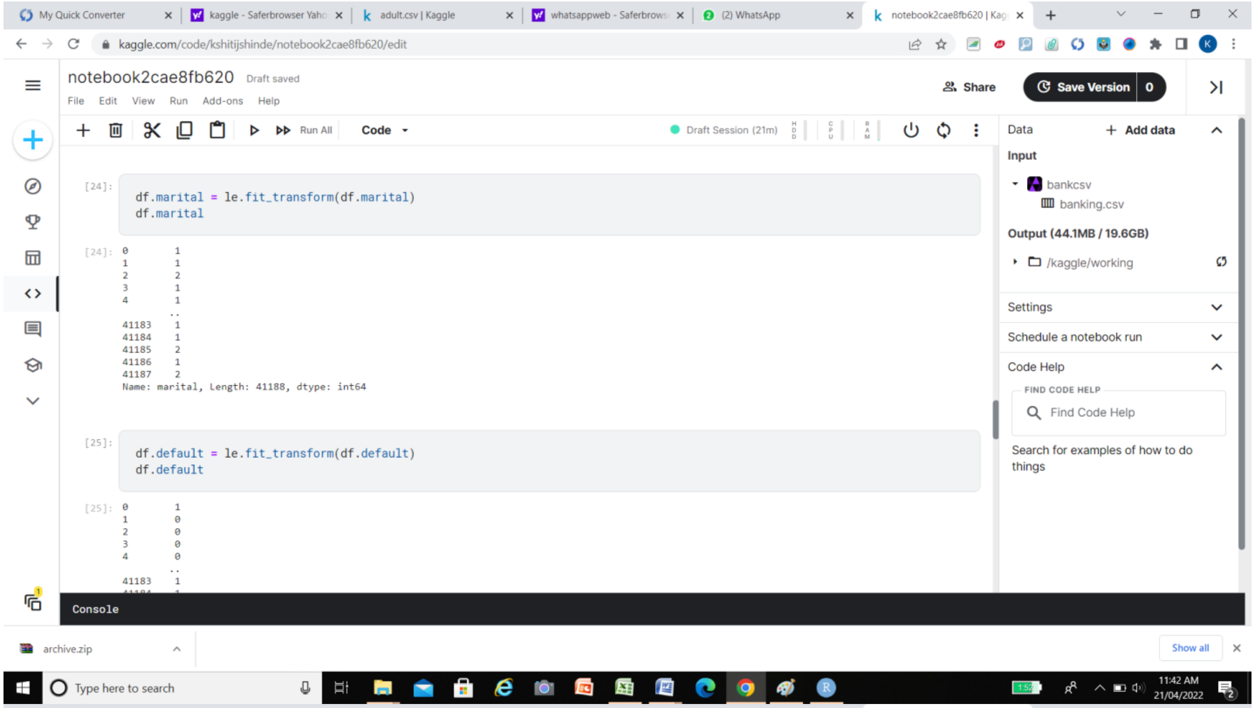
```
0        1
1        9
2        4
3        7
4        5
        ..
41183    5
41184    3
41185    0
41186    9
41187    8
Name: job, Length: 41188, dtype: int64
```

[24]:

```python
df.marital = le.fit_transform(df.marital)
df.marital
```

[24]:

```
0        1
1        1
2        2
3        1
```



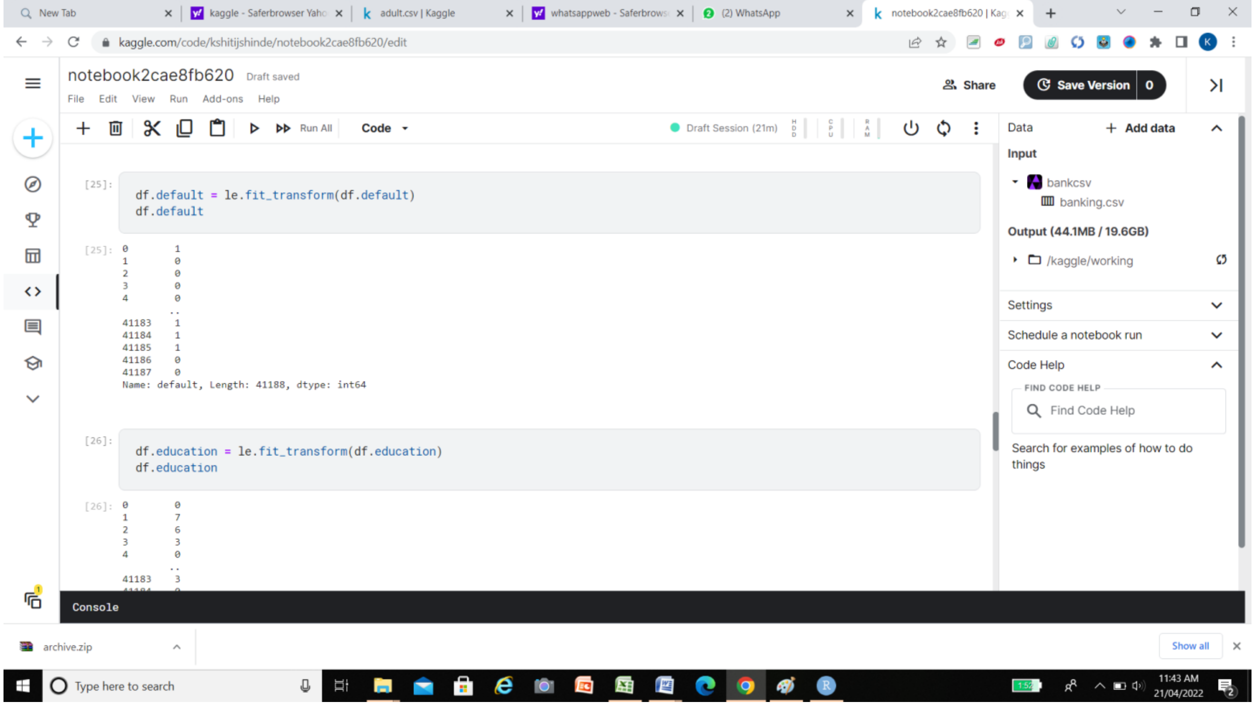
kaggle.com/code/kshitijshinde/notebook2cae8fb620/edit

[24]:
```
df.marital = le.fit_transform(df.marital)
df.marital
```

[24]:
```
0        1
1        1
2        2
3        1
4        1
        ..
41183    1
41184    1
41185    2
41186    1
41187    2
Name: marital, Length: 41188, dtype: int64
```

[25]:
```
df.default = le.fit_transform(df.default)
df.default
```

[25]:
```
0        1
1        0
2        0
3        0
4        0
        ..
41183    1
```



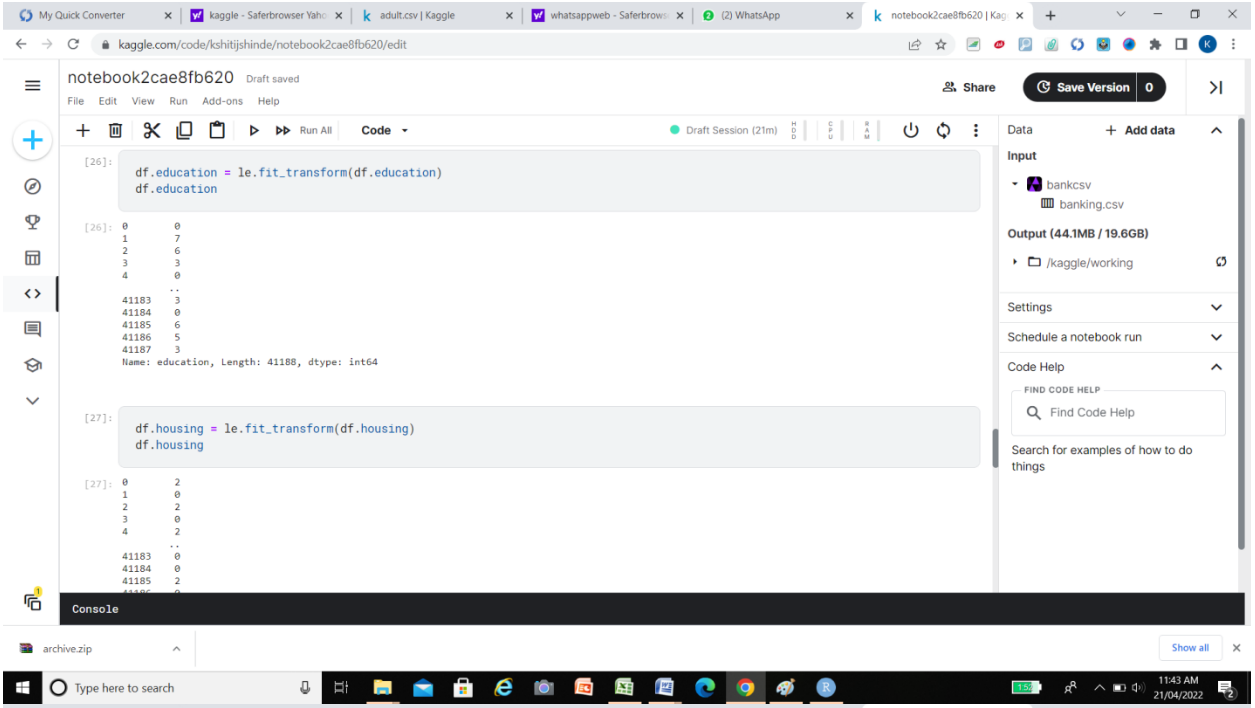
kaggle.com/code/kshitijshinde/notebook2cae8fb620/edit

[25]:
```
df.default = le.fit_transform(df.default)
df.default
```

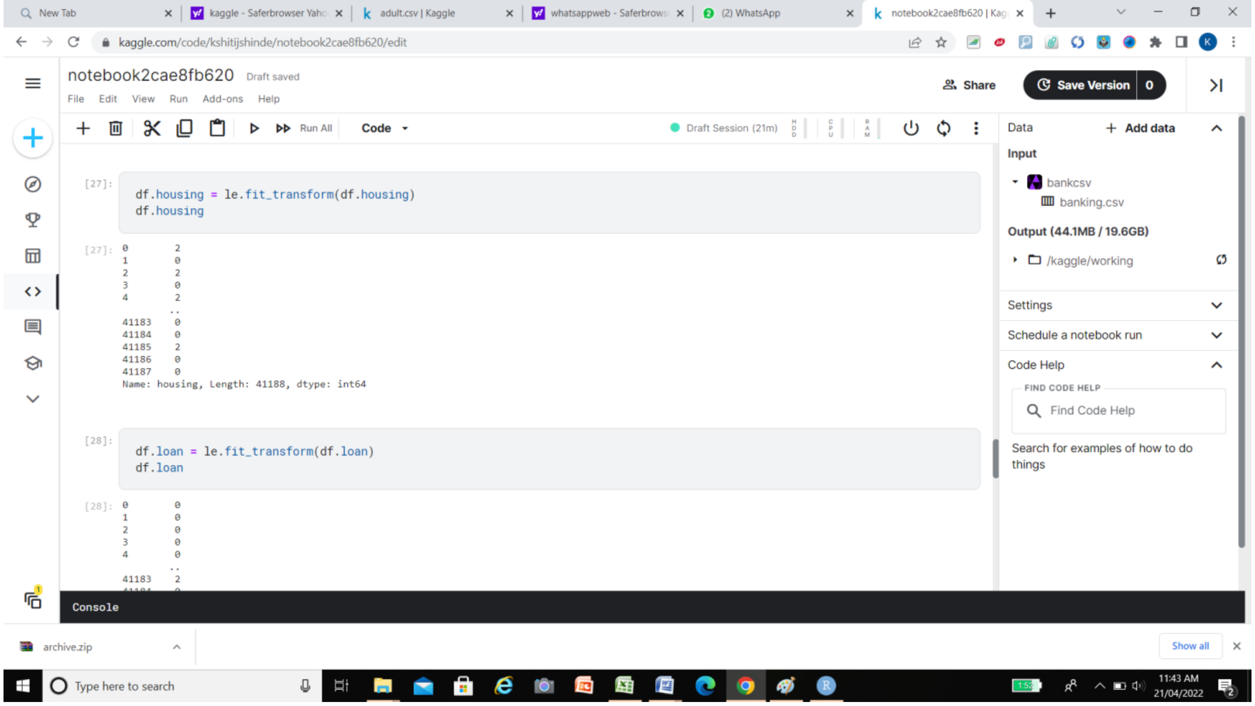
[25]:
```
0        1
1        0
2        0
3        0
4        0
        ..
41183    1
41184    1
41185    1
41186    0
41187    0
Name: default, Length: 41188, dtype: int64
```

[26]:
```
df.education = le.fit_transform(df.education)
df.education
```

[26]:
```
0        0
1        7
2        6
3        3
4        0
        ..
41183    3
```

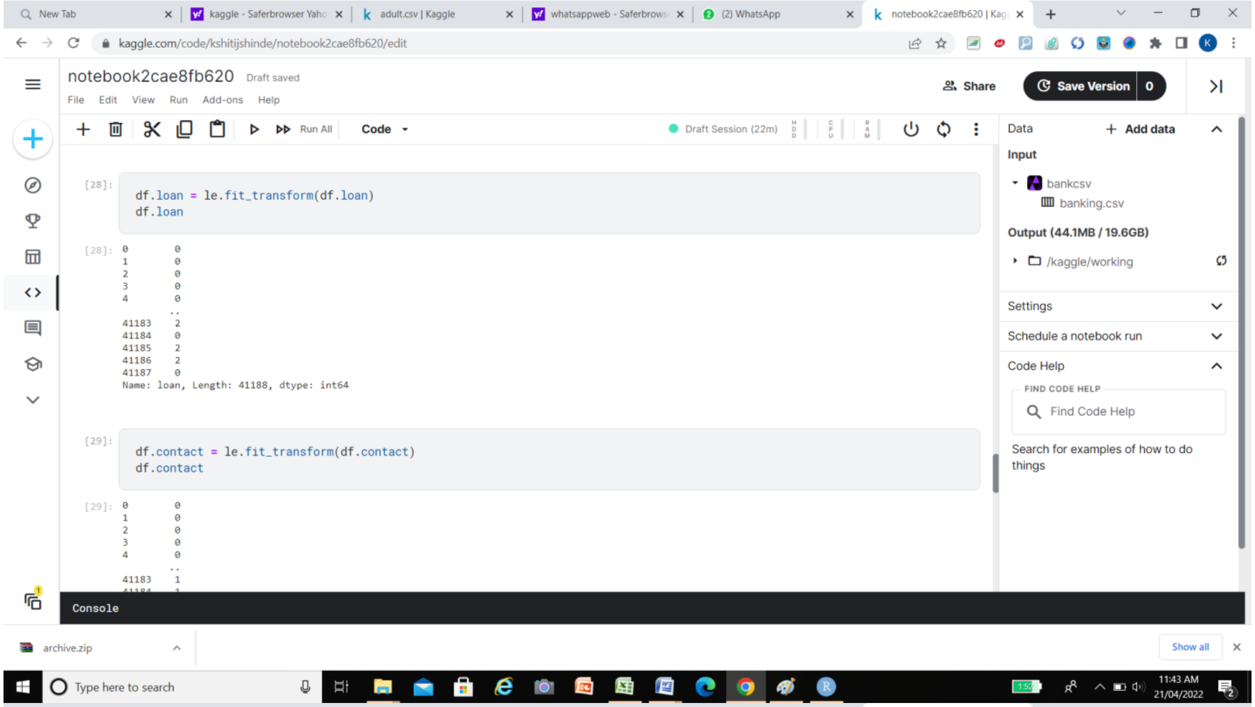


```
[26]:
df.education = le.fit_transform(df.education)
df.education

[26]: 0        0
      1        7
      2        6
      3        3
      4        0
              ..
      41183    3
      41184    0
      41185    6
      41186    5
      41187    3
      Name: education, Length: 41188, dtype: int64

[27]:
df.housing = le.fit_transform(df.housing)
df.housing

[27]: 0        2
      1        0
      2        2
      3        0
      4        2
              ..
      41183    0
      41184    0
      41185    2
```



```
[27]:
df.housing = le.fit_transform(df.housing)
df.housing
```

```
[27]: 0        2
      1        0
      2        2
      3        0
      4        2
              ..
      41183    0
      41184    0
      41185    2
      41186    0
      41187    0
      Name: housing, Length: 41188, dtype: int64

[28]:
df.loan = le.fit_transform(df.loan)
df.loan

[28]: 0        0
      1        0
      2        0
      3        0
      4        0
              ..
      41183    2
```

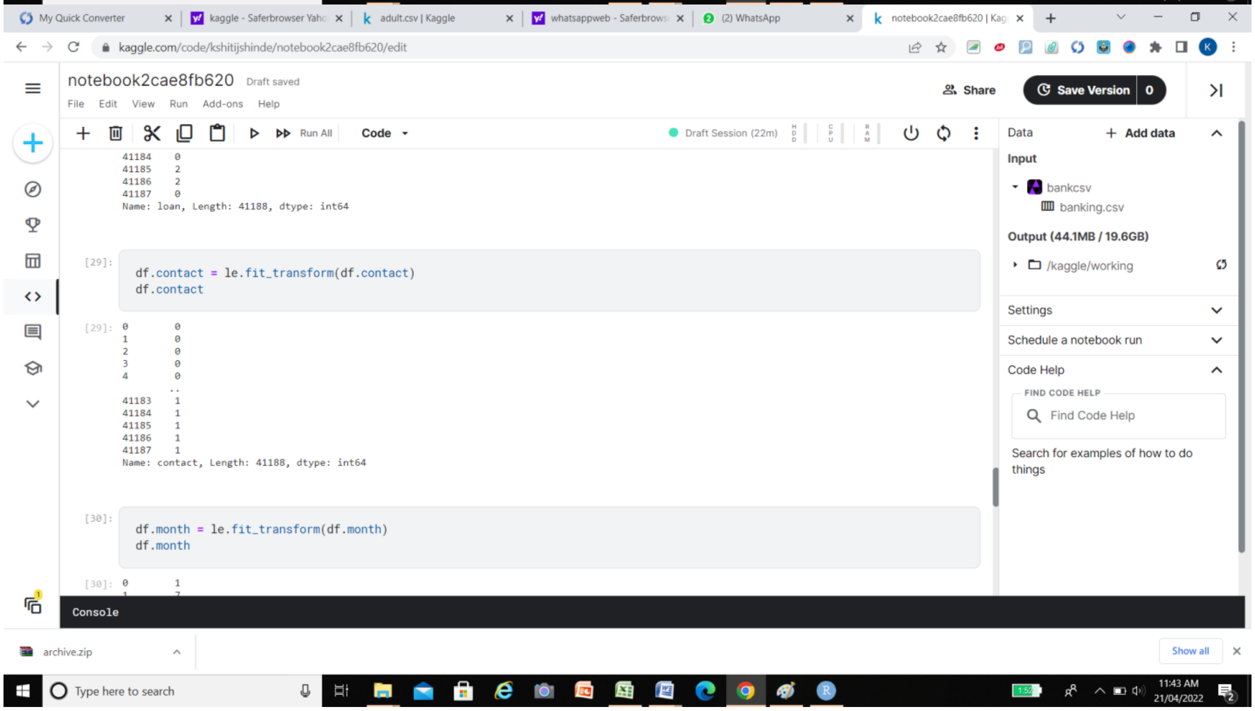


```
[28]:  df.loan = le.fit_transform(df.loan)
       df.loan
```

```
[28]: 0        0
      1        0
      2        0
      3        0
      4        0
              ..
      41183    2
      41184    0
      41185    2
      41186    2
      41187    0
      Name: loan, Length: 41188, dtype: int64
```

```
[29]:  df.contact = le.fit_transform(df.contact)
       df.contact
```

```
[29]: 0        0
      1        0
      2        0
      3        0
      4        0
              ..
      41183    1
```



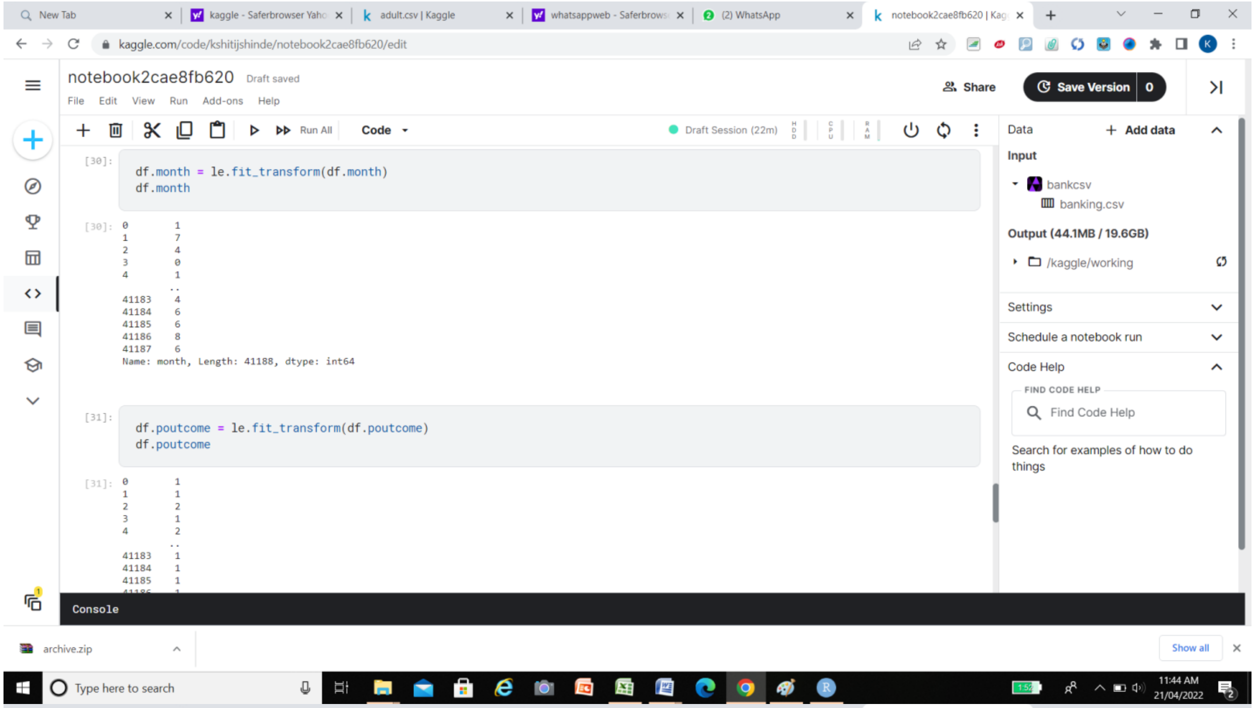
```
      41184    0
      41185    2
      41186    2
      41187    0
      Name: loan, Length: 41188, dtype: int64
```

```
[29]:  df.contact = le.fit_transform(df.contact)
       df.contact
```

```
[29]: 0        0
      1        0
      2        0
      3        0
      4        0
              ..
      41183    1
      41184    1
      41185    1
      41186    1
      41187    1
      Name: contact, Length: 41188, dtype: int64
```

```
[30]:  df.month = le.fit_transform(df.month)
       df.month
```

```
[30]: 0        1
```

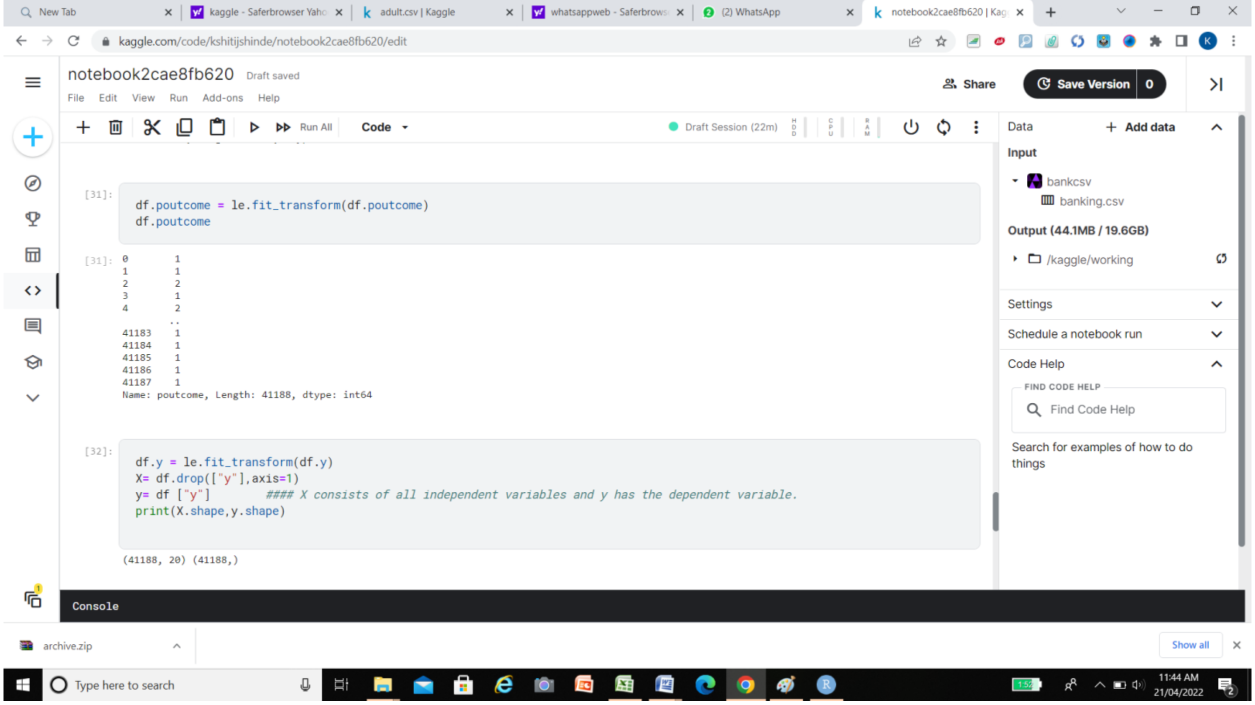


```
[30]:
df.month = le.fit_transform(df.month)
df.month
```

```
[30]: 0        1
1        7
2        4
3        0
4        1
        ..
41183    4
41184    6
41185    6
41186    8
41187    6
Name: month, Length: 41188, dtype: int64
```

```
[31]:
df.poutcome = le.fit_transform(df.poutcome)
df.poutcome
```

```
[31]: 0        1
1        1
2        2
3        1
4        2
        ..
41183    1
41184    1
41185    1
```

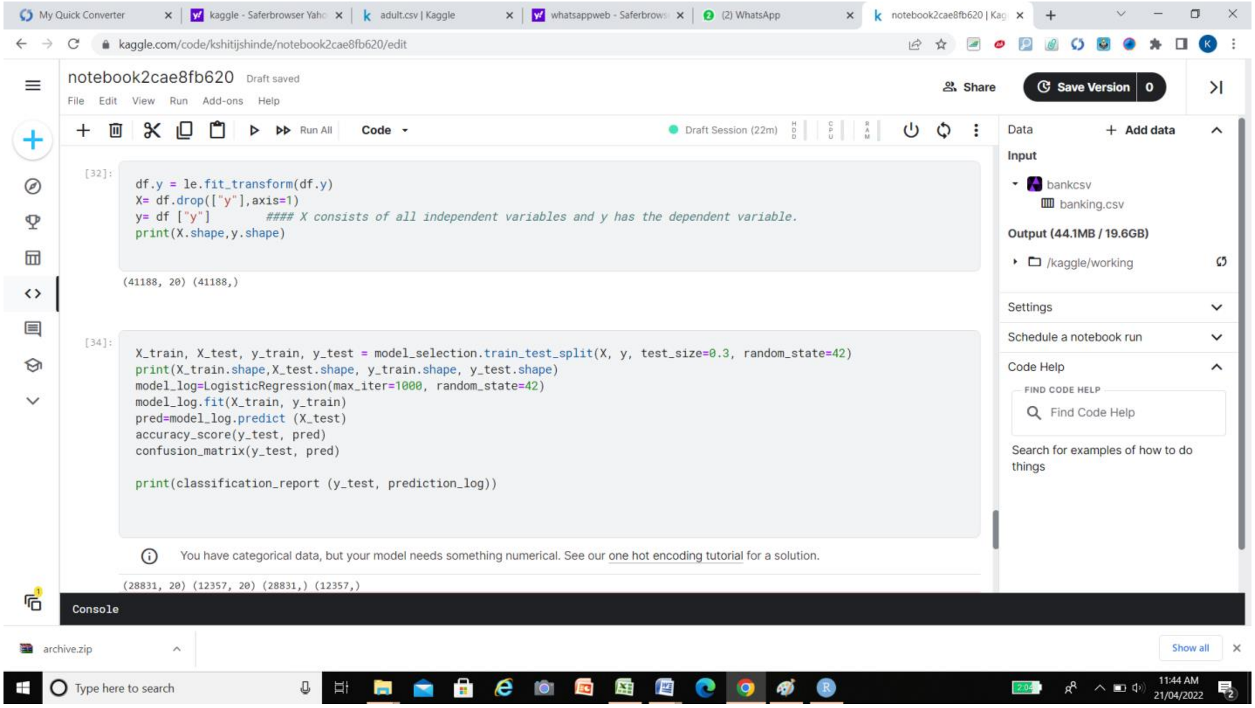


```
[31]:
df.poutcome = le.fit_transform(df.poutcome)
df.poutcome
```

```
[31]: 0        1
1        1
2        2
3        1
4        2
        ..
41183    1
41184    1
41185    1
41186    1
41187    1
Name: poutcome, Length: 41188, dtype: int64
```

```
[32]:
df.y = le.fit_transform(df.y)
X= df.drop(["y"],axis=1)
y= df ["y"]        #### X consists of all independent variables and y has the dependent variable.
print(X.shape,y.shape)
```

```
(41188, 20) (41188,)
```



[32]:

```
df.y = le.fit_transform(df.y)
X= df.drop(["y"],axis=1)
y= df ["y"]        #### X consists of all independent variables and y has the dependent variable.
print(X.shape,y.shape)
```

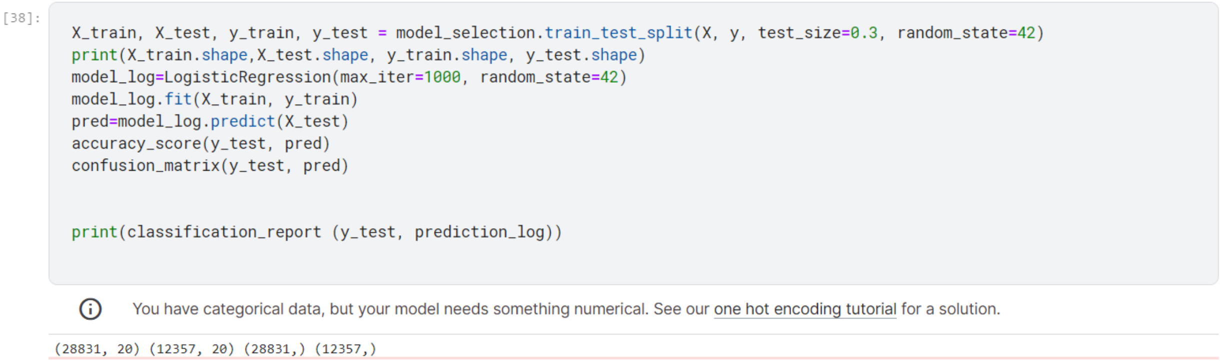
```
(41188, 20) (41188,)
```

[34]:

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.3, random_state=42)
print(X_train.shape,X_test.shape, y_train.shape, y_test.shape)
model_log=LogisticRegression(max_iter=1000, random_state=42)
model_log.fit(X_train, y_train)
pred=model_log.predict (X_test)
accuracy_score(y_test, pred)
confusion_matrix(y_test, pred)

print(classification_report (y_test, prediction_log))
```

ⓘ You have categorical data, but your model needs something numerical. See our one hot encoding tutorial for a solution.

```
(28831, 20) (12357, 20) (28831,) (12357,)
```



[38]:

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.3, random_state=42)
print(X_train.shape,X_test.shape, y_train.shape, y_test.shape)
model_log=LogisticRegression(max_iter=1000, random_state=42)
model_log.fit(X_train, y_train)
pred=model_log.predict(X_test)
accuracy_score(y_test, pred)
confusion_matrix(y_test, pred)


print(classification_report (y_test, prediction_log))
```

ⓘ You have categorical data, but your model needs something numerical. See our one hot encoding tutorial for a solution.

```
(28831, 20) (12357, 20) (28831,) (12357,)
```