task-2

February 6, 2024

[1]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

[2]:
```python
data = pd.read_csv('fraudTest.csv')
data = data.head(100000)
data
```

[2]:
```
       Unnamed: 0 trans_date_trans_time            cc_num  \
0               0   2020-06-21 12:14:25  2291163933867244
1               1   2020-06-21 12:14:33  3573030041201292
2               2   2020-06-21 12:14:53  3598215285024754
3               3   2020-06-21 12:15:15  3591919803438423
4               4   2020-06-21 12:15:17  3526826139003047
…             …                   …               …
99995       99995   2020-07-26 12:57:40  2242542703101233
99996       99996   2020-07-26 12:57:45  4482427013979020
99997       99997   2020-07-26 12:57:59  5580563567307107
99998       99998   2020-07-26 12:58:01  4294040533480516
99999       99999   2020-07-26 12:58:34  3577578023716568

                                  merchant       category    amt    first  \
0                     fraud_Kirlin and Sons  personal_care   2.86     Jeff
1                      fraud_Sporer-Keebler  personal_care  29.84   Joanne
2      fraud_Swaniawski, Nitzsche and Welch health_fitness  41.28   Ashley
3                         fraud_Haley Group       misc_pos  60.05    Brian
4                     fraud_Johnston-Casper         travel   3.19   Nathan
…                                       …              …    …      …
99995                       fraud_Kuhic Inc    grocery_pos  55.22   Samuel
99996                      fraud_Nienow PLC  entertainment  24.11   Leslie
99997     fraud_Romaguera, Wehner and Tromp      kids_pets  53.05  Stanley
99998                  fraud_Beier and Sons           home  26.93     Gail
99999               fraud_Wuckert-Goldner           home  36.46   Debbie
```

```
          last gender                           street  …      lat      long  \
0      Elliott      M              351 Darlene Green  …  33.9659  -80.9355
1     Williams      F                3638 Marsh Union  …  40.3207 -110.4360
2        Lopez      F           9333 Valentine Point  …  40.6729  -73.5365
3     Williams      M    32941 Krystal Mill Apt. 552  …  28.5697  -80.8191
4       Massey      M       5783 Evan Roads Apt. 465  …  44.2529  -85.0170
…          …    …                                …   …      …        …
99995  Jenkins      M  43235 Mckenzie Views Apt. 837  …  38.4921  -85.4524
99996     Ford      F           4938 Hatfield Course  …  38.8265  -82.1364
99997  Dickson      M                078 Alex Fields  …  39.9961  -79.7678
99998   Weaver      F               979 Stewart Lake  …  33.4130  -81.6900
99999   Hughes      F     0182 Owens Burgs Suite 480  …  41.0935  -81.0425

       city_pop                         job         dob  \
0        333497        Mechanical engineer  1968-03-19
1           302     Sales professional, IT  1990-01-17
2         34496          Librarian, public  1970-10-21
3         54767               Set designer  1987-07-25
4          1126         Furniture designer  1955-07-06
…           …                          …           …
99995       564        Pensions consultant  1996-04-10
99996       642  Building services engineer  1946-08-30
99997      1946         Charity fundraiser  1990-06-21
99998      2206       Biomedical scientist  1986-12-31
99999      2644       Engineer, biomedical  1983-08-25

                              trans_num   unix_time  merch_lat  merch_long  \
0      2da90c7d74bd46a0caf3777415b3ebd3  1371816865  33.986391  -81.200714
1      324cc204407e99f51b0d6ca0055005e7  1371816873  39.450498 -109.960431
2      c81755dbbbea9d5c77f094348a7579be  1371816893  40.495810  -74.196111
3      2159175b9efe66dc301f149d3d5abf8c  1371816915  28.812398  -80.883061
4      57ff021bd3f328f8738bb535c302a31b  1371816917  44.959148  -85.884734
…                                   …         …          …          …
99995  ee5e83124a95fb735f9b8a7566d08cc3  1374843460  37.934354  -85.979408
99996  5a4ec3ca3dd6c1d6c5d1882904d4688c  1374843465  38.360258  -81.656605
99997  51e122398ac61aeacde592249ad0a45d  1374843479  39.024515  -80.428413
99998  87be5ba3d74ce9267731f7ab4d035aa9  1374843481  34.172849  -82.476306
99999  ad1e02803cfc9f2da3a078c1cbff1a53  1374843514  41.252843  -80.234852

       is_fraud
0             0
1             0
2             0
3             0
4             0
…          …
```

```
99995        0
99996        0
99997        0
99998        0
99999        0

[100000 rows x 23 columns]
```

[3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Unnamed: 0             100000 non-null  int64
 1   trans_date_trans_time  100000 non-null  object
 2   cc_num                 100000 non-null  int64
 3   merchant               100000 non-null  object
 4   category               100000 non-null  object
 5   amt                    100000 non-null  float64
 6   first                  100000 non-null  object
 7   last                   100000 non-null  object
 8   gender                 100000 non-null  object
 9   street                 100000 non-null  object
 10  city                   100000 non-null  object
 11  state                  100000 non-null  object
 12  zip                    100000 non-null  int64
 13  lat                    100000 non-null  float64
 14  long                   100000 non-null  float64
 15  city_pop               100000 non-null  int64
 16  job                    100000 non-null  object
 17  dob                    100000 non-null  object
 18  trans_num              100000 non-null  object
 19  unix_time              100000 non-null  int64
 20  merch_lat              100000 non-null  float64
 21  merch_long             100000 non-null  float64
 22  is_fraud               100000 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 17.5+ MB
```

[4]:
```
data.dropna(subset=['unix_time', 'merch_lat', 'merch_long', 'is_fraud'],␣
↪inplace=True)
```

[5]:
```
null_values = data.isnull().sum()
print(null_values)
```

```
Unnamed: 0              0
trans_date_trans_time   0
cc_num                  0
merchant                0
category                0
amt                     0
first                   0
last                    0
gender                  0
street                  0
city                    0
state                   0
zip                     0
lat                     0
long                    0
city_pop                0
job                     0
dob                     0
trans_num               0
unix_time               0
merch_lat               0
merch_long              0
is_fraud                0
dtype: int64
```

[6]:

```
data.describe().T
```

[6]:

```
                count          mean           std           min           25%  \
Unnamed: 0   100000.0  4.999950e+04  2.886766e+04  0.000000e+00  2.499975e+04
cc_num       100000.0  4.134100e+17  1.303721e+18  6.041621e+10  1.800429e+14
amt          100000.0  6.928808e+01  1.526440e+02  1.000000e+00  9.650000e+00
zip          100000.0  4.881488e+04  2.684706e+04  1.257000e+03  2.629200e+04
lat          100000.0  3.854763e+01  5.064446e+00  2.002710e+01  3.466890e+01
long         100000.0 -9.020603e+01  1.370175e+01 -1.656723e+02 -9.679800e+01
city_pop     100000.0  8.887397e+04  3.016250e+05  2.300000e+01  7.430000e+02
unix_time    100000.0  1.373294e+09  8.750524e+05  1.371817e+09  1.372530e+09
merch_lat    100000.0  3.854606e+01  5.100006e+00  1.904232e+01  3.476584e+01
merch_long   100000.0 -9.020623e+01  1.371594e+01 -1.666463e+02 -9.689018e+01
is_fraud     100000.0  4.020000e-03  6.327622e-02  0.000000e+00  0.000000e+00

                      50%           75%           max
Unnamed: 0   4.999950e+04  7.499925e+04  9.999900e+04
cc_num       3.519233e+15  4.633065e+15  4.992346e+18
amt          4.732000e+01  8.305000e+01  1.314915e+04
zip          4.817400e+04  7.201100e+04  9.978300e+04
lat          3.937160e+01  4.194880e+01  6.568990e+01
long        -8.746925e+01 -8.017520e+01 -6.795030e+01
```

```
city_pop    2.408000e+03  1.968500e+04  2.906700e+06
unix_time   1.373239e+09  1.374066e+09  1.374844e+09
merch_lat   3.937735e+01  4.197288e+01  6.666936e+01
merch_long -8.742925e+01 -8.024994e+01 -6.695235e+01
is_fraud    0.000000e+00  0.000000e+00  1.000000e+00
```

[7]:
```
x = data.drop('is_fraud', axis=1)
y = data['is_fraud']
```

[8]:
```
data.columns
```

[8]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

[9]:
```
data['Unnamed: 0'],unnamd_name = pd.factorize(data['Unnamed: 0'])
unnamd_name
```

[9]:
```
Int64Index([    0,     1,     2,     3,     4,     5,     6,     7,     8,
                9,
            …
            99990, 99991, 99992, 99993, 99994, 99995, 99996, 99997, 99998,
            99999],
           dtype='int64', length=100000)
```

[10]:
```
data['cc_num'], cc_name = pd.factorize(data['cc_num'])
cc_name
```

[10]:
```
Int64Index([   2291163933867244,    3573030041201292,    3598215285024754,
               3591919803438423,    3526826139003047,      30407675418785,
                213180742685905,    3589289942931264,    3596357274378601,
               3546897637165774,
            …
            4358137750029944984,     180098888332620,    3531606252458308,
                 30373802285317,     377834944388609,    6526955903501879,
                  4026222041577,    4682744518117239,    3540416671210051,
                   503851367360],
           dtype='int64', length=911)
```

[11]:
```
data['trans_date_trans_time'],time_name=pd.
 ↪factorize(data['trans_date_trans_time'])
print(time_name)
```

```
Index(['2020-06-21 12:14:25', '2020-06-21 12:14:33', '2020-06-21 12:14:53',
       '2020-06-21 12:15:15', '2020-06-21 12:15:17', '2020-06-21 12:15:37',
```

```
 '2020-06-21 12:15:44', '2020-06-21 12:15:50', '2020-06-21 12:16:10',
 '2020-06-21 12:16:11',
 …
 '2020-07-26 12:56:44', '2020-07-26 12:56:48', '2020-07-26 12:57:03',
 '2020-07-26 12:57:09', '2020-07-26 12:57:17', '2020-07-26 12:57:40',
 '2020-07-26 12:57:45', '2020-07-26 12:57:59', '2020-07-26 12:58:01',
 '2020-07-26 12:58:34'],
dtype='object', length=98144)
```

[12]:
```
data['category'], category_name = pd.factorize(data['category'])
category_name
```

[12]:
```
Index(['personal_care', 'health_fitness', 'misc_pos', 'travel', 'kids_pets',
       'shopping_pos', 'food_dining', 'home', 'entertainment', 'shopping_net',
       'misc_net', 'grocery_pos', 'gas_transport', 'grocery_net'],
      dtype='object')
```

[13]:
```
data['merchant'], merchant_name = pd.factorize(data['merchant'])
merchant_name
```

[13]:
```
Index(['fraud_Kirlin and Sons', 'fraud_Sporer-Keebler',
       'fraud_Swaniawski, Nitzsche and Welch', 'fraud_Haley Group',
       'fraud_Johnston-Casper', 'fraud_Daugherty LLC', 'fraud_Romaguera Ltd',
       'fraud_Reichel LLC', 'fraud_Goyette, Howell and Collier',
       'fraud_Kilback Group',
       …
       'fraud_Rippin, Kub and Mann', 'fraud_Rempel PLC',
       'fraud_Leannon-Nikolaus', 'fraud_Monahan, Hermann and Johns',
       'fraud_Block-Hauck', 'fraud_Hagenes, Hermann and Stroman',
       'fraud_Hermann-Gaylord', 'fraud_Mante Group', 'fraud_Corwin-Gorczany',
       'fraud_McCullough Group'],
      dtype='object', length=693)
```

[14]:
```
data['amt'],amount = pd.factorize(data['amt'])
print(amount)
```

```
Float64Index([   2.86,   29.84,   41.28,   60.05,    3.19,   19.55,  133.93,
                10.37,    4.37,   66.54,
              …
              1074.88,  616.41,  225.08,  481.55,  176.05,  209.41,  168.81,
               747.31, 2606.86,  157.57],
             dtype='float64', length=20701)
```

[15]:
```
data['first'],first_name = pd.factorize(data['first'])
print(first_name)
```

```
Index(['Jeff', 'Joanne', 'Ashley', 'Brian', 'Nathan', 'Danielle', 'Kayla',
       'Paula', 'David', 'Samuel',
```

```
 …
 'Sean', 'Connor', 'Katelyn', 'Wesley', 'Sonya', 'Collin', 'Tommy',
 'Guy', 'Dennis', 'Bruce'],
dtype='object', length=339)
```

[16]:
```
data['last'],last_name = pd.factorize(data['last'])
print(last_name)
```

```
Index(['Elliott', 'Williams', 'Lopez', 'Massey', 'Evans', 'Sutton', 'Estrada',
       'Everett', 'Obrien', 'Jenkins',
       …
       'Prince', 'Chase', 'Heath', 'Copeland', 'Bridges', 'Raymond',
       'Davidson', 'Osborne', 'Webster', 'Freeman'],
      dtype='object', length=467)
```

[17]:
```
data['gender'],gender_name = pd.factorize(data['gender'])
print(gender_name)
```

```
Index(['M', 'F'], dtype='object')
```

[18]:
```
data['street'],street_name = pd.factorize(data['street'])
print(street_name)
```

```
Index(['351 Darlene Green', '3638 Marsh Union', '9333 Valentine Point',
       '32941 Krystal Mill Apt. 552', '5783 Evan Roads Apt. 465',
       '76752 David Lodge Apt. 064', '010 Weaver Land', '350 Stacy Glens',
       '4138 David Fall', '7921 Robert Port Suite 343',
       …
       '91542 Marissa Shores Apt. 053', '08469 Trujillo Forge',
       '7911 Campbell Crossing Apt. 725', '7538 Carrie Meadow Suite 574',
       '539 Underwood Divide', '7351 Cindy Well Suite 099',
       '204 Ashley Neck Apt. 169', '66035 Benjamin Villages',
       '44613 James Turnpike', '77686 Donald Bridge Apt. 711'],
      dtype='object', length=911)
```

[19]:
```
data['city'],city_name = pd.factorize(data['city'])
print(city_name)
```

```
Index(['Columbia', 'Altonah', 'Bellmore', 'Titusville', 'Falmouth',
       'Breesport', 'Carlotta', 'Spencer', 'Morrisdale', 'Prairie Hill',
       …
       'West Chazy', 'Oran', 'Springville', 'Stoneham', 'Claremont',
       'Pea Ridge', 'Preston', 'Syracuse', 'Rice', 'Grifton'],
      dtype='object', length=839)
```

[20]:
```
data['state'],state_name = pd.factorize(data['state'])
print(state_name)
```

```
Index(['SC', 'UT', 'NY', 'FL', 'MI', 'CA', 'SD', 'PA', 'TX', 'KY', 'WY', 'AL',
       'LA', 'GA', 'CO', 'OH', 'WI', 'VT', 'AR', 'NJ', 'IA', 'MD', 'MS', 'KS',
       'IL', 'MO', 'ME', 'TN', 'DC', 'AZ', 'MT', 'MN', 'OK', 'WA', 'WV', 'NM',
       'MA', 'NE', 'VA', 'ID', 'OR', 'IN', 'NC', 'NH', 'ND', 'CT', 'NV', 'HI',
       'RI', 'AK'],
      dtype='object')
```

[21]:
```
data['zip'],zip_name = pd.factorize(data['zip'])
print(zip_name)
```

```
Int64Index([29209, 84002, 11710, 32780, 49632, 14816, 95528, 57374, 16858,
            76678,
            …
            50664, 14141,  2180, 91711, 72751, 34120,  6365, 65354, 56367,
            28530],
           dtype='int64', length=900)
```

[22]:
```
data['lat'],lat_name = pd.factorize(data['lat'])
print(lat_name)
```

```
Float64Index([33.9659, 40.3207, 40.6729, 28.5697, 44.2529, 42.1939,  40.507,
              43.7557, 41.0001, 31.6591,
              …
              42.7012,   42.52, 42.4828, 34.1092, 36.4539, 26.3304, 41.5224,
              38.6547, 45.7364, 35.3757],
             dtype='float64', length=898)
```

[23]:
```
data['long'],long_name = pd.factorize(data['long'])
print(long_name)
```

```
Float64Index([          -80.9355,           -110.436,           -73.5365,
                        -80.8191, -85.01700000000001,           -76.7361,
                       -123.9743,           -97.5936,           -78.2357,
                        -96.8094,
              …
                        -73.5112,           -92.0762,           -71.0978,
                       -117.7183,            -94.118,           -81.5871,
                        -71.9934,           -92.8929,           -94.1658,
                        -77.4193],
             dtype='float64', length=898)
```

[24]:
```
data['city_pop'],city_name = pd.factorize(data['city_pop'])
print(city_name)
```

```
Int64Index([333497,    302,  34496,  54767,   1126,    520,   1139,    343,
              3688,    263,
            …
               533,   4778,   7728,  21437,  35705,   6434,   4720,    628,
```

```
             6263,    7332],
           dtype='int64', length=825)
```

[25]:
```python
data['job'],job_name = pd.factorize(data['job'])
print(job_name)
```

```
Index(['Mechanical engineer', 'Sales professional, IT', 'Librarian, public',
       'Set designer', 'Furniture designer', 'Psychotherapist',
       'Therapist, occupational', 'Development worker, international aid',
       'Advice worker', 'Barrister',
       …
       'English as a foreign language teacher', 'Hydrogeologist',
       'Medical technical officer', 'Charity officer', 'Administrator, arts',
       'Occupational therapist', 'Solicitor, Scotland', 'Sports administrator',
       'Artist', 'Engineer, water'],
      dtype='object', length=476)
```

[26]:
```python
data['dob'],dob_name = pd.factorize(data['dob'])
print(dob_name)
```

```
Index(['1968-03-19', '1990-01-17', '1970-10-21', '1987-07-25', '1955-07-06',
       '1991-10-13', '1951-01-15', '1972-03-05', '1973-05-27', '1956-05-30',
       …
       '1972-10-05', '1959-03-30', '1964-06-25', '1956-05-15', '1967-08-28',
       '1950-12-14', '1977-05-18', '1961-12-18', '1944-05-30', '1957-06-27'],
      dtype='object', length=897)
```

[27]:
```python
data['trans_num'],trans_num_name = pd.factorize(data['trans_num'])
print(trans_num_name)
```

```
Index(['2da90c7d74bd46a0caf3777415b3ebd3', '324cc204407e99f51b0d6ca0055005e7',
       'c81755dbbbea9d5c77f094348a7579be', '2159175b9efe66dc301f149d3d5abf8c',
       '57ff021bd3f328f8738bb535c302a31b', '798db04aaceb4febd084f1a7c404da93',
       '17003d7ce534440eadb10c4750e020e5', '8be473af4f05fc6146ea55ace73e7ca2',
       '71a1da150d1ce510193d7622e08e784e', 'a7915132c7c4240996ba03a47f81e3bd',
       …
       '25a711596ed1f84583ba2bb392160185', '863193c1fb20c4cbf4be6ecbeeb70df4',
       '7d816d579f113588a3c19312185d938e', 'e38ec5afc0262abe41c9016bb3c5d52e',
       'c9eefe145c133ffb7128d5dc8704992b', 'ee5e83124a95fb735f9b8a7566d08cc3',
       '5a4ec3ca3dd6c1d6c5d1882904d4688c', '51e122398ac61aeacde592249ad0a45d',
       '87be5ba3d74ce9267731f7ab4d035aa9', 'ad1e02803cfc9f2da3a078c1cbff1a53'],
      dtype='object', length=100000)
```

[28]:
```python
data['unix_time'],unix_time_name = pd.factorize(data['unix_time'])
print(unix_time_name)
```

```
Int64Index([1371816865, 1371816873, 1371816893, 1371816915, 1371816917,
            1371816937, 1371816944, 1371816950, 1371816970, 1371816971,
```

```
               …
               1374843404, 1374843408, 1374843423, 1374843429, 1374843437,
               1374843460, 1374843465, 1374843479, 1374843481, 1374843514],
              dtype='int64', length=98144)
```

[29]:
```python
data['merch_lat'],merch_lat_name = pd.factorize(data['merch_lat'])
print(merch_lat_name)
```

```
Float64Index([33.986391, 39.450498,  40.49581, 28.812398, 44.959148, 41.747157,
              41.499458, 44.495498, 41.546067, 31.782919,
              …
              41.199374, 36.871195, 34.565993, 40.778345, 39.832395, 37.934354,
              38.360258, 39.024515, 34.172849, 41.252843],
             dtype='float64', length=99686)
```

[30]:
```python
data['merch_long'],merch_long_name = pd.factorize(data['merch_long'])
print(merch_long_name)
```

```
Float64Index([        -81.200714,        -109.960431,         -74.196111,
                      -80.883061,         -85.884734,         -77.584197,
                     -124.888729,         -97.728453,         -78.120238,
                      -96.366185,
              …
              -78.61323399999999, -79.93264599999999,        -117.249455,
                       -87.57594,        -120.868586, -85.97940799999999,
                      -81.656605,         -80.428413, -82.47630600000001,
                      -80.234852],
             dtype='float64', length=99863)
```

[31]:
```python
data['is_fraud'],is_fraud_name = pd.factorize(data['is_fraud'])
print(is_fraud_name)
```

```
Int64Index([0, 1], dtype='int64')
```

[32]:
```python
x=data.iloc[:,0:-1]
y=data.iloc[:,-1]
```

[33]:
```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=42)
```

[34]:
```python
models = {
    ('random_model' ,RandomForestClassifier()),
    ('logistic_model', LogisticRegression()),
    ('decision_model', DecisionTreeClassifier()),

}
```

[35]:
```python
models
```

[35]: {('decision_model', DecisionTreeClassifier()),
 ('logistic_model', LogisticRegression()),
 ('random_model', RandomForestClassifier())}

[36]:
```
results = pd.DataFrame(columns=['Model', 'Accuracy_score'])
```

[37]:
```
for model_name , model in models:
    model.fit(x_train,y_train)
    prediction = model.predict(x_test)
    accuracy_score_models = accuracy_score(y_test,prediction )
    results = results.append({'Model':model_name, 'Accuracy_score':
 ↪accuracy_score_models},
                             ignore_index=True)
    classification_report_model = classification_report(prediction, y_test)
    confusion_matrix_model = confusion_matrix(prediction, y_test)
    print(f'{model_name} : Model_name')
    print(f'confusion matrix:\n {confusion_matrix_model}')
    print(f'classification report:\n {classification_report_model}')

print(results)
```

```
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\Admin\AppData\Local\Temp\ipykernel_9920\3868633738.py:5: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({'Model':model_name,
'Accuracy_score':accuracy_score_models},
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1471: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
logistic_model : Model_name
confusion matrix:
 [[19911    89]
 [    0     0]]
classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     20000
           1       0.00      0.00      0.00         0

    accuracy                           1.00     20000
   macro avg       0.50      0.50      0.50     20000
weighted avg       1.00      1.00      1.00     20000
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_9920\3868633738.py:5: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({'Model':model_name,
'Accuracy_score':accuracy_score_models},
```

```
random_model : Model_name
confusion matrix:
 [[19911    20]
 [    0    69]]
classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     19931
           1       0.78      1.00      0.87        69

    accuracy                           1.00     20000
   macro avg       0.89      1.00      0.94     20000
weighted avg       1.00      1.00      1.00     20000
```

```
decision_model : Model_name
confusion matrix:
 [[19879    36]
 [   32    53]]
classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     19915
```

```
           1       0.60      0.62      0.61        85

    accuracy                           1.00     20000
   macro avg       0.80      0.81      0.80     20000
weighted avg       1.00      1.00      1.00     20000

            Model  Accuracy_score
0  logistic_model         0.99555
1    random_model         0.99900
2  decision_model         0.99660
C:\Users\Admin\AppData\Local\Temp\ipykernel_9920\3868633738.py:5: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({'Model':model_name,
'Accuracy_score':accuracy_score_models},
```

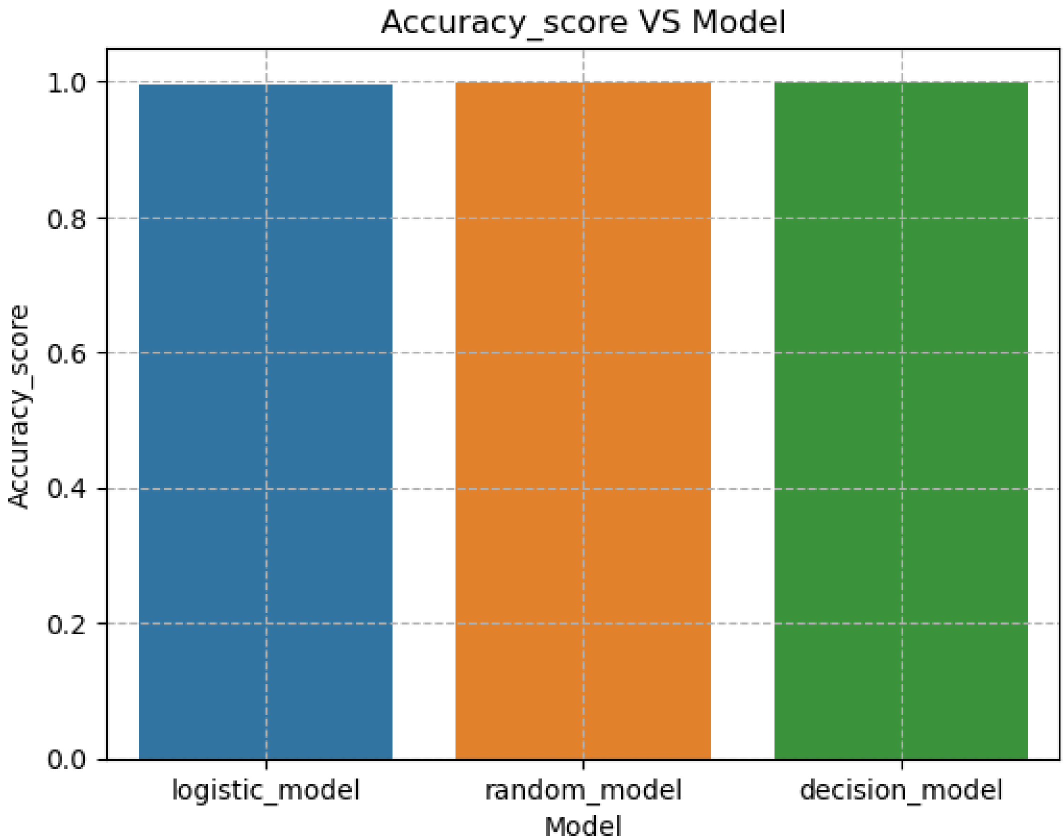
[38]:
```python
new_data = pd.DataFrame(results)
new_data
```

[38]:

	Model	Accuracy_score
0	logistic_model	0.99555
1	random_model	0.99900
2	decision_model	0.99660

[41]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(x='Model', y='Accuracy_score', data=new_data)
plt.grid(linestyle='--')
plt.title('Accuracy_score VS Model')
```

[41]:
```
Text(0.5, 1.0, 'Accuracy_score VS Model')
```


Accuracy_score VS Model
Accuracy_score
1.0
0.8
0.6
0.4
0.2
0.0
logistic_model
random_model
decision_model
Model

```
[ ]:
```

```
[ ]:
```