Code Way Internship Task 2

Name :- Sampurna Dey

Task Name :- Build a model to detect fraudulent credit card transactions


```
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
```

```
data = pd.read_csv('/content/fraudTest.csv')
data = data.head(100000)
data
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M	351 Darlene Green
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F	3638 Marsh Union
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F	9333 Valentine Point
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Williams	M	32941 Krystal Mill Apt. 552
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Massey	M	5783 Evan Roads Apt. 465
...	...	...	...	...	...	...	...	...	...	...
11692	11692	2020-06-24 22:21:35	4653178848915023204	fraud_Beier LLC	entertainment	11.58	Robert	Hall	M	371 Anthony Trail Suite 354
11693	11693	2020-06-24 22:22:53	4681699462969	fraud_Willms, Kris and Bergnaum	shopping_pos	9.84	Joseph	Gonzalez	M	319 Wendy Fort Suite 179
11694	11694	2020-06-24 22:23:44	4003989662068504	fraud_Torphy-Kertzmann	health_fitness	37.98	Chris	White	M	98897 Bennett Lodge
11695	11695	2020-06-24 22:24:26	639046421587	fraud_Abbott-Rogahn	entertainment	75.39	Dylan	Bonilla	M	2497 John Motorway Suite 922
11696	11696	2020-06-24 22:24:58	6011367958204270	fraud_Torphy-Goyette	shopping_pos	8.37	Tammy	Ayers	F	1652 James Mews

11697 rows × 23 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11697 entries, 0 to 11696
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Unnamed: 0             11697 non-null  int64  
 1   trans_date_trans_time  11697 non-null  object 
 2   cc_num                 11697 non-null  int64  
 3   merchant               11697 non-null  object 
 4   category               11697 non-null  object 
 5   amt                    11697 non-null  float64
 6   first                  11697 non-null  object 
 7   last                   11697 non-null  object 
 8   gender                 11697 non-null  object 
 9   street                 11697 non-null  object 
 10  city                   11697 non-null  object 
 11  state                  11697 non-null  object 
 12  zip                    11697 non-null  int64  
 13  lat                    11697 non-null  float64
 14  long                   11697 non-null  float64
 15  city_pop               11697 non-null  int64  
 16  job                    11697 non-null  object 
 17  dob                    11697 non-null  object 
 18  trans_num              11697 non-null  object 
 19  unix_time              11696 non-null  float64
 20  merch_lat              11696 non-null  float64
 21  merch_long             11696 non-null  float64
 22  is_fraud               11696 non-null  float64
dtypes: float64(7), int64(4), object(12)
memory usage: 2.1+ MB
```

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

```
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
```

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

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	11696.0	5.847500e+03	3.376489e+03	0.000000e+00	2.923750e+03	5.847500e+03	8.771250e+03	1.169500e+04
cc_num	11696.0	4.065880e+17	1.295855e+18	6.041621e+10	1.800429e+14	3.518669e+15	4.634956e+15	4.992346e+18
amt	11696.0	6.624624e+01	1.240199e+02	1.000000e+00	9.200000e+00	4.377500e+01	7.977000e+01	3.396840e+03
zip	11696.0	4.858389e+04	2.670791e+04	1.257000e+03	2.623700e+04	4.803400e+04	7.201100e+04	9.978300e+04
lat	11696.0	3.849113e+01	5.112459e+00	2.002710e+01	3.450910e+01	3.934260e+01	4.201440e+01	6.568990e+01
long	11696.0	-9.003008e+01	1.357620e+01	-1.656723e+02	-9.661840e+01	-8.696570e+01	-8.017520e+01	-6.795030e+01
city_pop	11696.0	9.472538e+04	3.222138e+05	2.300000e+01	7.820000e+02	2.691000e+03	2.163500e+04	2.906700e+06
unix_time	11696.0	1.371949e+09	7.917888e+04	1.371817e+09	1.371885e+09	1.371941e+09	1.372010e+09	1.372113e+09
merch_lat	11696.0	3.847797e+01	5.144570e+00	1.916345e+01	3.462704e+01	3.933204e+01	4.199723e+01	6.595173e+01
merch_long	11696.0	-9.002884e+01	1.359273e+01	-1.664644e+02	-9.659576e+01	-8.723814e+01	-8.027051e+01	-6.712295e+01
is_fraud	11696.0	2.479480e-03	4.973473e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00

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

```
 data.columns
```

```
     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')
```

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

```
     Int64Index([    0,     1,     2,     3,     4,     5,     6,     7,     8,
                     9,
                 ...
                 11686, 11687, 11688, 11689, 11690, 11691, 11692, 11693, 11694,
                 11695],
                dtype='int64', length=11696)
```

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

```
     Int64Index([   2291163933867244,    3573030041201292,    3598215285024754,
                    3591919803438423,    3526826139003047,      30407675418785,
                     213180742685905,    3589289942931264,    3596357274378601,
                    3546897637165774,
                 ...
                      36581538659449, 4666314527820883145,    3530503964418806,
                    4874006077381178,      36360452125889, 4358137750029944984,
                     180098888332620,    3531606252458308,      30373802285317,
                     377834944388609],
                dtype='int64', length=906)
```

```
 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',
            ...
            '2020-06-24 22:16:34', '2020-06-24 22:17:27', '2020-06-24 22:18:33',
            '2020-06-24 22:20:01', '2020-06-24 22:21:04', '2020-06-24 22:21:25',
            '2020-06-24 22:21:35', '2020-06-24 22:22:53', '2020-06-24 22:23:44',
            '2020-06-24 22:24:26'],
           dtype='object', length=11448)
```

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

```
     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')
```

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

```
     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)
```

```
 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,
```

```
                 ...
                 173.77,  35.35,  23.01, 156.78,  39.25, 373.86, 790.54, 723.23,
                  95.85,  66.53],
                dtype='float64', length=7280)
```

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

```
     Index(['Jeff', 'Joanne', 'Ashley', 'Brian', 'Nathan', 'Danielle', 'Kayla',
            'Paula', 'David', 'Samuel',
            ...
            'Bobby', 'Sean', 'Connor', 'Katelyn', 'Wesley', 'Sonya', 'Collin',
            'Tommy', 'Guy', 'Dennis'],
           dtype='object', length=338)
```

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

```
     Index(['Elliott', 'Williams', 'Lopez', 'Massey', 'Evans', 'Sutton', 'Estrada',
            'Everett', 'Obrien', 'Jenkins',
            ...
            'Franco', 'Bush', 'Prince', 'Chase', 'Heath', 'Copeland', 'Bridges',
            'Raymond', 'Davidson', 'Osborne'],
           dtype='object', length=465)
```

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

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

```
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',
            ...
            '957 Miller Falls', '5812 Ramos Oval Suite 598',
            '0638 Fred Ramp Suite 086', '89297 Wilson Green Suite 601',
            '766 Potter Well', '91542 Marissa Shores Apt. 053',
            '08469 Trujillo Forge', '7911 Campbell Crossing Apt. 725',
            '7538 Carrie Meadow Suite 574', '539 Underwood Divide'],
           dtype='object', length=906)
```

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

```
     Index(['Columbia', 'Altonah', 'Bellmore', 'Titusville', 'Falmouth',
            'Breesport', 'Carlotta', 'Spencer', 'Morrisdale', 'Prairie Hill',
            ...
            'Apison', 'Ravenna', 'Palmdale', 'Moscow', 'West Chazy', 'Oran',
            'Springville', 'Stoneham', 'Claremont', 'Pea Ridge'],
           dtype='object', length=835)
```

```
 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')
```

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

```
     Int64Index([29209, 84002, 11710, 32780, 49632, 14816, 95528, 57374, 16858,
                 76678,
                 ...
                 37302, 68869, 93552, 52760, 12992, 50664, 14141,  2180, 91711,
                 72751],
                dtype='int64', length=895)
```

```
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,
                   ...
                   35.0149, 41.0233, 34.5715, 41.5646,  44.797, 42.7012,   42.52,
                   42.4828, 34.1092, 36.4539],
                  dtype='float64', length=893)
```

```
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,
                   ...
                             -117.281,           -85.0164,           -98.9041,
                            -118.0231,           -91.0859,           -73.5112,
                             -92.0762,           -71.0978,          -117.7183,
                              -94.118],
                  dtype='float64', length=893)
```

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

```
     Int64Index([333497,    302,  34496,  54767,   1126,    520,   1139,    343,
                   3688,    263,
                  ...
                    641,   3730,   2202, 171170,    533,   4778,   7728,  21437,
                  35705,   6434],
                 dtype='int64', length=821)
```

```
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',
            ...
            'Economist', 'English as a foreign language teacher', 'Hydrogeologist',
            'Medical technical officer', 'Charity officer', 'Administrator, arts',
            'Occupational therapist', 'Solicitor, Scotland', 'Sports administrator',
            'Artist'],
           dtype='object', length=475)
```

```
 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',
            ...
            '1991-01-28', '1974-06-21', '1963-06-04', '1990-01-13', '1954-06-14',
            '1972-10-05', '1959-03-30', '1964-06-25', '1956-05-15', '1967-08-28'],
           dtype='object', length=892)
```

```
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',
            ...
            'adba949e4f3bf15bd2afa13f10171f0f', 'c31d08702a00c82e1cfc788d84ae1539',
            '15132fed7cb161dd480c99fdb17a36ab', '3b9324a9b482a0d962bce935b1f5d659',
            '419a6d22fd97e9ff543b9d95a108dc80', '4c07571a5bfa3054cd053212bb13c5d5',
            '80b30ed7d4cd22bbebc334c46c5a256f', '6fe69162ab8ae8d4ab7dc5bea60c165e',
            '15d5bc996a54050116076299d24373de', '8f570cf5c23fe6d79194a6ea402970b9'],
           dtype='object', length=11696)
```

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

```
     Float64Index([1371816865.0, 1371816873.0, 1371816893.0, 1371816915.0,
                   1371816917.0, 1371816937.0, 1371816944.0, 1371816950.0,
                   1371816970.0, 1371816971.0,
                   ...
                   1372112194.0, 1372112247.0, 1372112313.0, 1372112401.0,
```

```
                 1372112464.0, 1372112485.0, 1372112495.0, 1372112573.0,
                 1372112624.0, 1372112666.0],
                dtype='float64', length=11448)
```

```
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,
              ...
                         40.6071,           31.42292,          31.195173,
                       41.292895,          41.754307,          49.504031,
                       36.914435,          35.709589, 33.249666999999995,
                       40.149897],
              dtype='float64', length=11694)
```

```
 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,
               ...
               -105.780717, -103.779166, -105.851257,  -76.174377,  -73.363945,
               -118.243803,  -82.552026,  -85.437614,  -87.785931,  -79.834239],
              dtype='float64', length=11696)
```

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

```
Float64Index([0.0, 1.0], dtype='float64')
```

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

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
 models = {
('random_model' ,RandomForestClassifier()),
('logistic_model', LogisticRegression()),
('decision_model', DecisionTreeClassifier()),
}
```

```
models
```

```
{('decision_model', DecisionTreeClassifier()),
 ('logistic_model', LogisticRegression()),
 ('random_model', RandomForestClassifier())}
```

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

```
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)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
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(
    decision_model : Model_name
    confusion matrix:
     [[2334    0]
     [   2    4]]
    classification report:
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00      2334
               1       1.00      0.67      0.80         6

        accuracy                           1.00      2340
       macro avg       1.00      0.83      0.90      2340
    weighted avg       1.00      1.00      1.00      2340

                Model  Accuracy_score
    0  decision_model        0.999145
    <ipython-input-42-f8944942f396>:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future
      results = results.append({'Model':model_name, 'Accuracy_score':accuracy_score_models},
```

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

	Model	Accuracy_score
0	decision_model	0.999145

```
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')
```

```
Text(0.5, 1.0, 'Accuracy_score VS Model')
```