```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files archive.zip

- **archive.zip**(application/zip) - 186385561 bytes, last modified: 11/1/2023 - 1% done

```
import zipfile
with zipfile.ZipFile('archive.zip', 'r') as zip_ref:
    zip_ref.extractall('/content/archive')
```

```
data = pd.read_csv('/content/PS_20174392719_1491204439457_log.csv')
data.head()
```

```
data.info()
```

```
data.describe()
```

```
obj = (data.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:", len(object_cols))

int_ = (data.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:", len(num_cols))

fl = (data.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:", len(fl_cols))
```

```
sns.countplot(x='type', data=data)
```

```
sns.barplot(x='type', y='amount', data=data)
```

```
data['isFraud'].value_counts()
```

```
plt.figure(figsize=(15, 6))
sns.distplot(data['step'], bins=50)
```

```
plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(),
            cmap='BrBG',
            fmt='.2f',
            linewidths=2,
            annot=True)
```

```
type_new = pd.get_dummies(data['type'], drop_first=True)
data_new = pd.concat([data, type_new], axis=1)
data_new.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703

```
X = data_new.drop(['isFraud', 'type', 'nameOrig', 'nameDest'], axis=1)
y = data_new['isFraud']
```

```
X.shape, y.shape
```

```
((97225, 11), (97225,))
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)
```

```
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score as ras
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```
models = [LogisticRegression(), XGBClassifier(),
        SVC(kernel='rbf', probability=True),
        RandomForestClassifier(n_estimators=7,
                               criterion='entropy',
                               random_state=7)]

for i in range(len(models)):
    models[i].fit(X_train, y_train)
    print(f'{models[i]} : ')
```

```
    train_preds = models[i].predict_proba(X_train)[:, 1]
    print('Training Accuracy : ', ras(y_train, train_preds))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: 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(
LogisticRegression() :
Training Accuracy :  0.9412209029475906
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...) :
Training Accuracy :  1.0
SVC(probability=True) :
Training Accuracy :  0.9222025711325856
RandomForestClassifier(criterion='entropy', n_estimators=7, random_state=7) :
Training Accuracy :  0.9999969124905377
```

```
from sklearn.impute import SimpleImputer

# Create an imputer
imputer = SimpleImputer(strategy='mean')  # You can change the strategy

# Fit the imputer on your training data
imputer.fit(X_test)
y_test = y_test.fillna(y_test.mean())



# Transform the test data, filling missing values
X_test_imputed = imputer.transform(X_test)
Y_test_imputed = imputer.transform(y_test)
y_preds = models[i].predict_proba(X_test_imputed)[:, 1]
```

```
print('Validation Accuracy : ', ras(Y_test_imputed, y_preds))
print()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-40-ed365f372c5b> in <cell line: 14>()
     12 # Transform the test data, filling missing values
     13 X_test_imputed = imputer.transform(X_test)
---> 14 Y_test_imputed = imputer.transform(y_test)
     15 y_preds = models[i].predict_proba(X_test_imputed)[:, 1]
     16
```

```
from sklearn.metrics import roc_auc_score

# Assuming you have a trained 'model', replace 'models' with 'model' in your code
y_preds = models.predict_proba(X_test)[:, 1]
print('Validation ROC AUC Score: ', roc_auc_score(y_test, y_preds))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-29-96d2925331f8> in <cell line: 4>()
      2
      3 # Assuming you have a trained 'model', replace 'models' with 'model' in your code
----> 4 y_preds = models.predict_proba(X_test)[:, 1]
      5 print('Validation ROC AUC Score: ', roc_auc_score(y_test, y_preds))

AttributeError: 'list' object has no attribute 'predict_proba'

SEARCH STACK OVERFLOW
```

```
from sklearn.metrics import roc_auc_score

# models is a list of trained models
roc_auc_scores = []

for model in models:
    y_preds = model.predict_proba(X_test)[:, 1]
    roc_auc = roc_auc_score(y_test, y_preds)
    roc_auc_scores.append(roc_auc)

print('Validation ROC AUC Scores:', roc_auc_scores)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-30-c20689b728d7> in <cell line: 6>()
      5
      6 for model in models:
----> 7     y_preds = model.predict_proba(X_test)[:, 1]
      8     roc_auc = roc_auc_score(y_test, y_preds)
      9     roc_auc_scores.append(roc_auc)

                                  5 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype, estimator_name, input_
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162
    163

ValueError: Input X contains NaN.
LogisticRegression does not accept missing values encoded as NaN natively. For supervised learning, you might want to consider sklearn.er
NaNs natively. Alternatively, it is possible to preprocess the data, for instance by using an imputer transformer in a pipeline or drop
find a list of all estimators that handle NaN values at the following page: https://scikit-learn.org/stable/modules/impute.html#estimato

SEARCH STACK OVERFLOW
```

```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(models[1], X_test, y_test)
plt.show()
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-23-e18bb2b94739> in <cell line: 1>()
----> 1 from sklearn.metrics import plot_confusion_matrix
      2
      3 plot_confusion_matrix(models[1], X_test, y_test)
      4 plt.show()

ImportError: cannot import name 'plot_confusion_matrix' from 'sklearn.metrics' (/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

```
y_preds = models[i].predict_proba(X_test)[:, 1]
print('Validation Accuracy : ', ras(y_test, y_preds))
print()
```

```
LogisticRegression() :
Training Accuracy :  0.9412209029475906
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: 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(
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-24-0afa5273d512> in <cell line: 7>()
     12         print('Training Accuracy : ', ras(y_train, train_preds))
     13
---> 14         y_preds = models[i].predict_proba(X_test)[:, 1]
     15         print('Validation Accuracy : ', ras(y_test, y_preds))
     16         print()

                                    5 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype, estimator_name, input_
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162
    163

ValueError: Input X contains NaN.
LogisticRegression does not accept missing values encoded as NaN natively. For supervised learning, you might want to consider sklearn.er
NaNs natively. Alternatively, it is possible to preprocess the data, for instance by using an imputer transformer in a pipeline or drop
find a list of all estimators that handle NaN values at the following page: https://scikit-learn.org/stable/modules/impute.html#estimato
```

SEARCH STACK OVERFLOW