1)DETAIL ABOUT PROBLEM

The dataset includes reservation information for hotel. There are 32 columns in dataset. Reservation status in the dataset show that Check-Out, Canceled or No-Show. In this study, target is reservation status. Reservation status are classified according to features in the datatese and using some machine learning algorithms.

In [ ]:
```
import pandas as pd
data = pd.read_csv('hotel_bookings.csv')
```

2) DESCRIPTION OF DATASET

The data shape. There are 119390 rows and 32 columns.

In [ ]:
```
data.shape
```

Out[ ]:
```
(119390, 32)
```

This is variable types of each columsn.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   hotel                           119390 non-null  object 
 1   is_canceled                     119390 non-null  int64  
 2   lead_time                       119390 non-null  int64  
 3   arrival_date_year               119390 non-null  int64  
 4   arrival_date_month              119390 non-null  object 
 5   arrival_date_week_number        119390 non-null  int64  
 6   arrival_date_day_of_month       119390 non-null  int64  
 7   stays_in_weekend_nights         119390 non-null  int64  
 8   stays_in_week_nights            119390 non-null  int64  
 9   adults                          119390 non-null  int64  
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64  
 12  meal                            119390 non-null  object 
 13  country                         118902 non-null  object 
 14  market_segment                  119390 non-null  object 
 15  distribution_channel            119390 non-null  object 
 16  is_repeated_guest               119390 non-null  int64  
 17  previous_cancellations          119390 non-null  int64  
 18  previous_bookings_not_canceled  119390 non-null  int64  
 19  reserved_room_type              119390 non-null  object 
 20  assigned_room_type              119390 non-null  object 
 21  booking_changes                 119390 non-null  int64  
 22  deposit_type                    119390 non-null  object 
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64  
 26  customer_type                   119390 non-null  object 
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64  
 29  total_of_special_requests       119390 non-null  int64  
 30  reservation_status              119390 non-null  object 
 31  reservation_status_date         119390 non-null  object 
```

```
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

In [ ]:
```
data.head()
```

Out[ ]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	
3	Resort Hotel	0	13	2015	July	27	
4	Resort Hotel	0	14	2015	July	27	

5 rows × 32 columns

In [ ]:
```
data.describe()
```

Out[ ]:

	is_canceled	lead_time	arrival_date_year	arrival_date_week_number	arrival_date_day_of_month	stays
count	119390.000000	119390.000000	119390.000000	119390.000000	119390.000000	
mean	0.370416	104.011416	2016.156554	27.165173	15.798241	
std	0.482918	106.863097	0.707476	13.605138	8.780829	
min	0.000000	0.000000	2015.000000	1.000000	1.000000	
25%	0.000000	18.000000	2016.000000	16.000000	8.000000	
50%	0.000000	69.000000	2016.000000	28.000000	16.000000	
75%	1.000000	160.000000	2017.000000	38.000000	23.000000	
max	1.000000	737.000000	2017.000000	53.000000	31.000000	

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        4
babies                          0
meal                            0
country                       488
market_segment                  0
distribution_channel            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
```

```
reserved_room_type                  0
assigned_room_type                  0
booking_changes                     0
deposit_type                        0
agent                           16340
company                        112593
days_in_waiting_list                0
customer_type                       0
adr                                 0
required_car_parking_spaces         0
total_of_special_requests           0
reservation_status                  0
reservation_status_date             0
dtype: int64
```

There are missing value in the dataset. We should fill this missing value or delete them.

In [ ]:
```
data = data.drop("company",axis=1)
```

In [ ]:
```
data = data.drop("agent",axis=1)
```

In [ ]:
```
data = data.dropna()
```

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
hotel                           0
is_canceled                     0
lead_time                       0
arrival_date_year               0
arrival_date_month              0
arrival_date_week_number        0
arrival_date_day_of_month       0
stays_in_weekend_nights         0
stays_in_week_nights            0
adults                          0
children                        0
babies                          0
meal                            0
country                         0
market_segment                  0
distribution_channel            0
is_repeated_guest               0
previous_cancellations          0
previous_bookings_not_canceled  0
reserved_room_type              0
assigned_room_type              0
booking_changes                 0
deposit_type                    0
days_in_waiting_list            0
customer_type                   0
adr                             0
required_car_parking_spaces     0
total_of_special_requests       0
reservation_status              0
reservation_status_date         0
dtype: int64
```

In [ ]:
```
data.duplicated().sum()
```

Out[ ]:
```
31984
```

There are duplicate date, we can delete them

In [ ]:
```
data = data.drop_duplicates(keep='first').reset_index(drop=True)
data.duplicated().sum()
```

Out[ ]:
```
0
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(86914, 30)
```

In [ ]:
```
data['hotel'].value_counts()
#there are two type of hotel
```

Out[ ]:
```
City Hotel      53404
Resort Hotel    33510
Name: hotel, dtype: int64
```

In [ ]:
```
data['arrival_date_month'].value_counts()
#there are twelve type of arrival date month
```

Out[ ]:
```
August       11229
July         10020
May           8341
April         7869
June          7752
March         7457
October       6883
September     6657
February      6040
December      5080
November      4950
January       4636
Name: arrival_date_month, dtype: int64
```

In [ ]:
```
data['meal'].value_counts()
#there are five type of meal
```

Out[ ]:
```
BB           67540
SC            9473
HB            9054
Undefined      488
FB             359
Name: meal, dtype: int64
```

In [ ]:
```
data['country'].value_counts()
#there are 177 type of country
```

Out[ ]:
```
PRT    27436
GBR    10431
FRA     8837
ESP     7250
DEU     5385
       ...
MMR        1
BFA        1
CYM        1
MLI        1
KHM        1
Name: country, Length: 177, dtype: int64
```

In [ ]:
```
data['market_segment'].value_counts()
#there are seven type of market segment
```

Out[ ]:
```
Online TA        51534
Offline TA/TO    13849
Direct           11645
Groups            4936
```

```
Corporate         4025
Complementary      698
Aviation           227
Name: market_segment, dtype: int64
```

In [ ]:
```
data['distribution_channel'].value_counts()
#there are five type of distribution channel
```

Out[ ]:
```
TA/TO        69010
Direct       12825
Corporate     4897
GDS            181
Undefined        1
Name: distribution_channel, dtype: int64
```

In [ ]:
```
data['reserved_room_type'].value_counts()
#there are ten type of reserved room
```

Out[ ]:
```
A    56166
D    17370
E     6009
F     2816
G     2041
B      995
C      914
H      596
L        6
P        1
Name: reserved_room_type, dtype: int64
```

In [ ]:
```
data['assigned_room_type'].value_counts()
#there are twelve type of assigned room
```

Out[ ]:
```
A    46131
D    22275
E     7126
F     3607
G     2484
C     2144
B     1816
H      702
I      351
K      276
L        1
P        1
Name: assigned_room_type, dtype: int64
```

In [ ]:
```
data['deposit_type'].value_counts()
#there are three type of deposit
```

Out[ ]:
```
No Deposit    85770
Non Refund     1037
Refundable      107
Name: deposit_type, dtype: int64
```

In [ ]:
```
data['customer_type'].value_counts()
#there are four type of customer
```

Out[ ]:
```
Transient          71554
Transient-Party    11684
Contract            3139
Group                537
Name: customer_type, dtype: int64
```

In [ ]:
```
data['reservation_status'].value_counts()
#there are three type of reservation status
```

Out[ ]:
```
Check-Out    62931
Canceled     22973
No-Show       1010
Name: reservation_status, dtype: int64
```

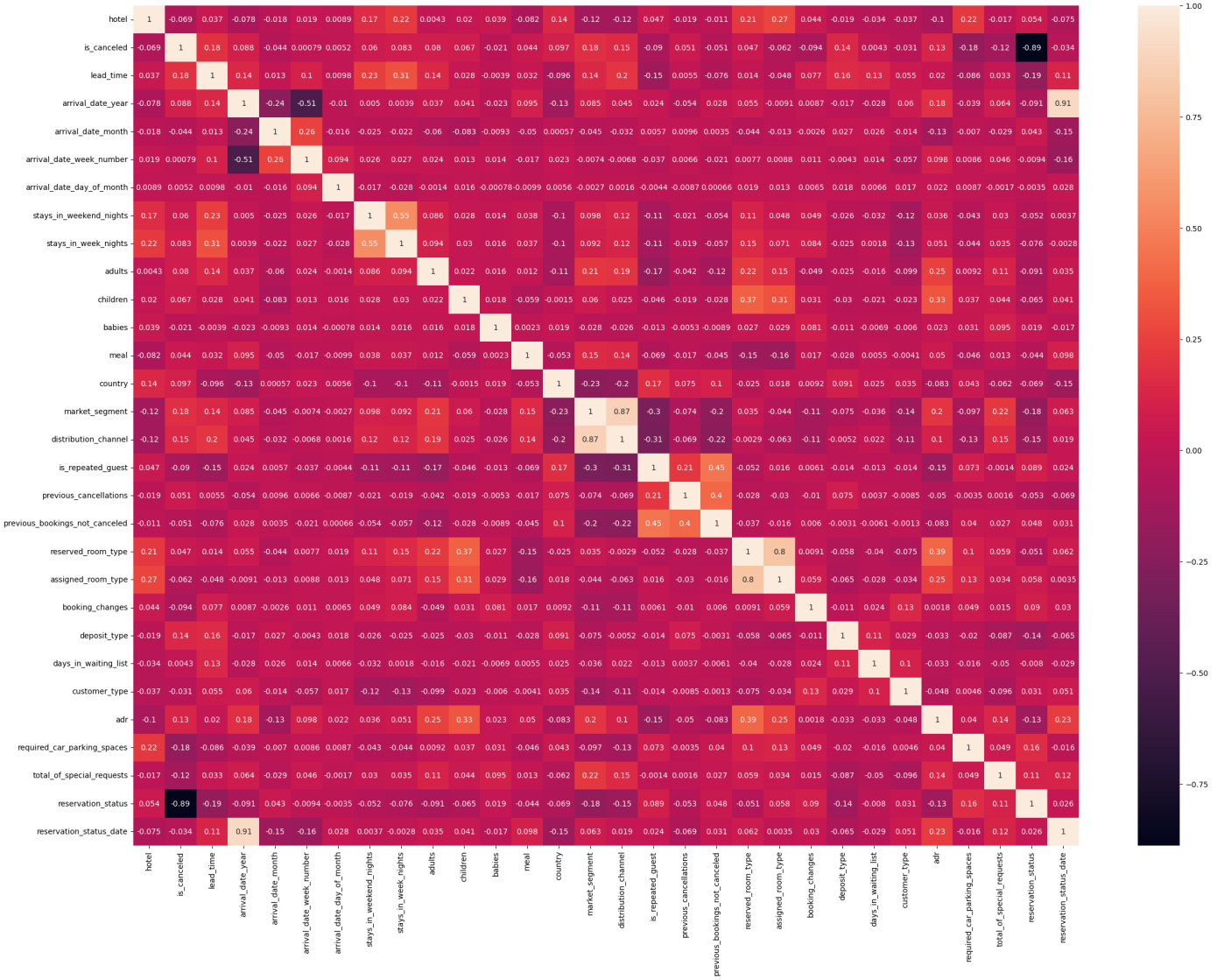
In [ ]:
```
data['reservation_status_date'].value_counts()
#there are 926 type of reservation status date
```

Out[ ]:
```
2016-02-14    210
2017-05-25    204
2015-10-21    199
2016-10-06    195
2016-03-28    195
             ...
2015-02-12      1
2015-02-19      1
2015-02-27      1
2015-02-24      1
2015-04-18      1
Name: reservation_status_date, Length: 926, dtype: int64
```

In [ ]:
```
#we use label encoding for string values
from sklearn import preprocessing
labelencoder=preprocessing.LabelEncoder()

data['hotel']      = labelencoder.fit_transform(data['hotel'])
data['arrival_date_month']    = labelencoder.fit_transform(data['arrival_date_month'])
data['meal']    = labelencoder.fit_transform(data['meal'])
data['country']       = labelencoder.fit_transform(data['country'])
data['market_segment']    = labelencoder.fit_transform(data['market_segment'])
data['distribution_channel']    = labelencoder.fit_transform(data['distribution_channel'])
data['reserved_room_type']       = labelencoder.fit_transform(data['reserved_room_type'])
data['assigned_room_type']    = labelencoder.fit_transform(data['assigned_room_type'])
data['deposit_type']    = labelencoder.fit_transform(data['deposit_type'])
data['customer_type']       = labelencoder.fit_transform(data['customer_type'])
data['reservation_status']    = labelencoder.fit_transform(data['reservation_status'])
data['reservation_status_date']    = labelencoder.fit_transform(data['reservation_status_
```

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

data.corr()
fig,ax=plt.subplots(figsize=(28,20))
sns.heatmap(data.corr(),annot=True)
```

Out[ ]:
```
<Axes: >
```



3)MACHINE LEARNING MODELS

Reservation status is output for classification models.

In [ ]:
```
y=data.pop('reservation_status')
y
```

Out[ ]:
```
0         1
1         1
2         1
3         1
4         1
         ..
86909     1
86910     1
86911     1
86912     1
86913     1
Name: reservation_status, Length: 86914, dtype: int64
```

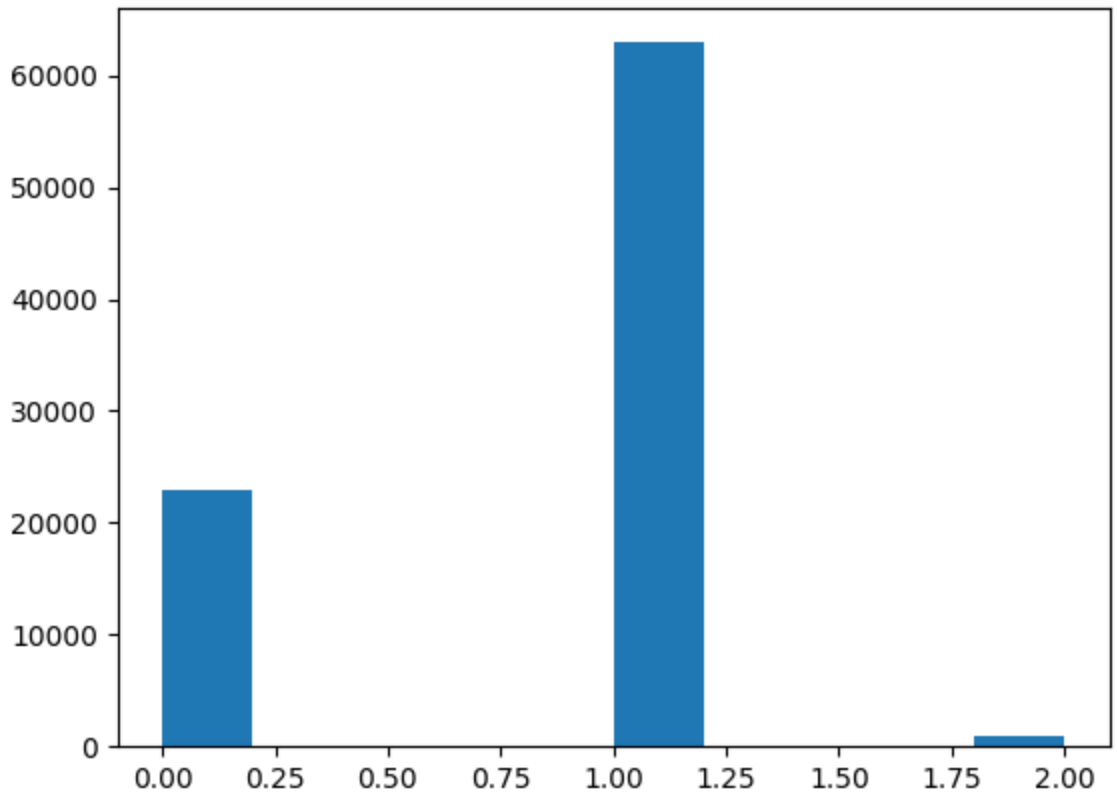
In [ ]:
```
X=data.to_numpy()
X
```

Out[ ]:
```
array([[  1.,   0., 342., ...,   0.,   0., 121.],
       [  1.,   0., 737., ...,   0.,   0., 121.],
       [  1.,   0.,   7., ...,   0.,   0., 122.],
       ...,
       [  0.,   0.,  34., ...,   0.,   4., 920.],
```

```
       [  0.,   0., 109., ...,   0.,   0., 920.],
       [  0.,   0., 205., ...,   0.,   2., 920.]])
```

In [ ]:
```
plt.hist(y)
#0:canceled
#1:check-out
#2:no-show
```

Out[ ]:
```
(array([22973.,     0.,     0.,     0.,     0., 62931.,     0.,     0.,
            0.,  1010.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,stratify=y,test_size=0.2)
```

Decision Tree Classifier

In [ ]:
```
import sklearn.tree as tree
clf=tree.DecisionTreeClassifier(class_weight='balanced')
clf.fit(X_train,y_train)
y_predict=clf.predict(X_test)
```

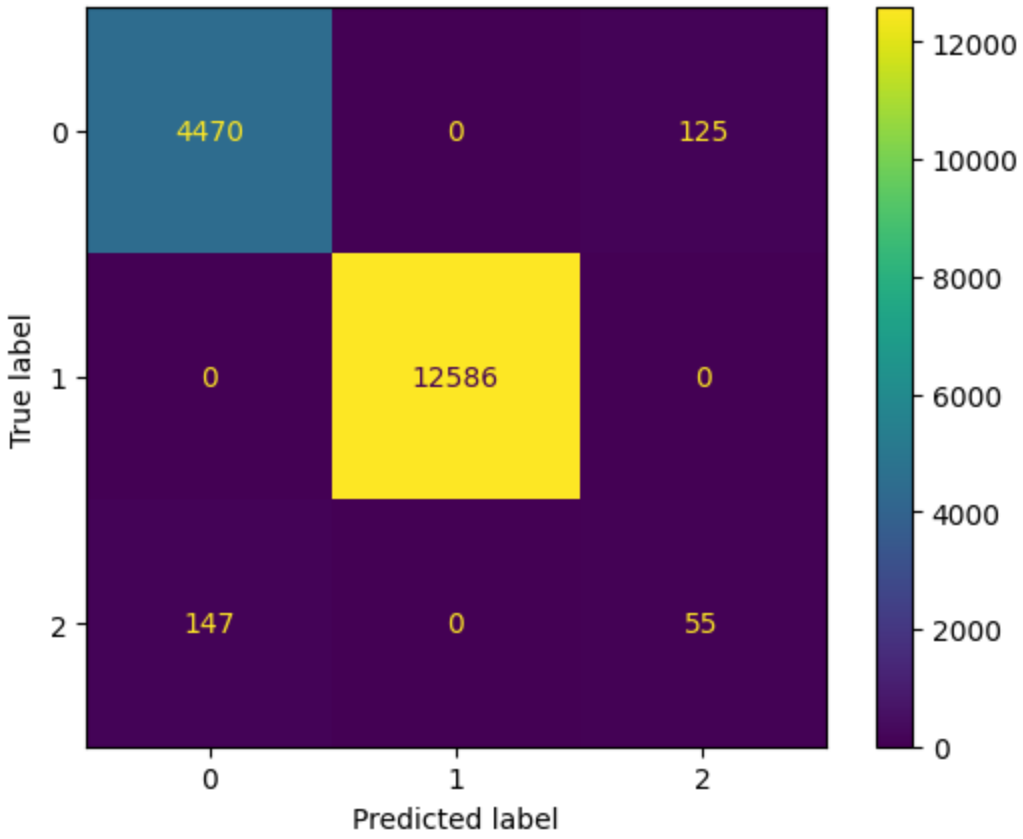
In [ ]:
```
pip install --upgrade scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
ic/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages
(1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-learn) (3.1.0)
```

In [ ]:
```
from sklearn.metrics import classification_report
```

```
from sklearn.metrics import ConfusionMatrixDisplay

print(classification_report(y_test,y_predict))
ConfusionMatrixDisplay.from_estimator(clf,X_test,y_test)
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      4595
           1       1.00      1.00      1.00     12586
           2       0.31      0.27      0.29       202

    accuracy                           0.98     17383
   macro avg       0.76      0.75      0.75     17383
weighted avg       0.98      0.98      0.98     17383
```



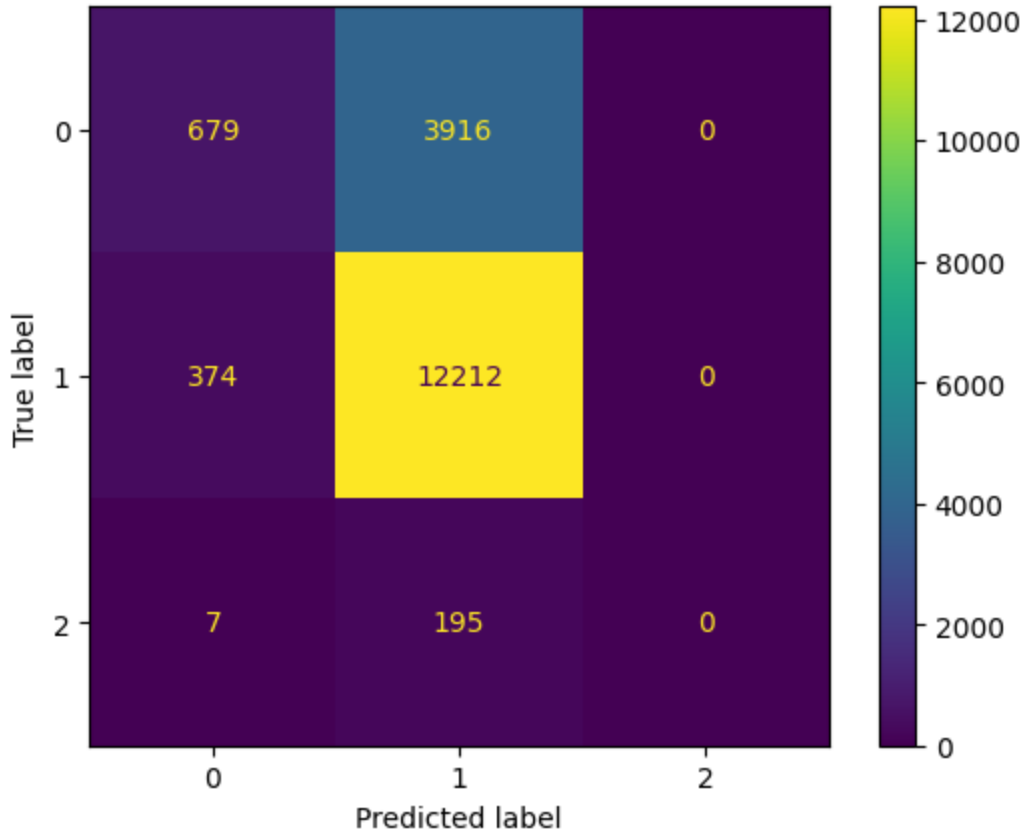
Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression
clf=LogisticRegression()
clf.fit(X_train,y_train)
y_predict=clf.predict(X_test)

print(classification_report(y_test,y_predict))
ConfusionMatrixDisplay.from_estimator(clf, X_test, y_test)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: 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(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Undefin
```

```
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: Undefin
edMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wi
th no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

	precision	recall	f1-score	support
0	0.64	0.15	0.24	4595
1	0.75	0.97	0.84	12586
2	0.00	0.00	0.00	202
accuracy			0.74	17383
macro avg	0.46	0.37	0.36	17383
weighted avg	0.71	0.74	0.68	17383



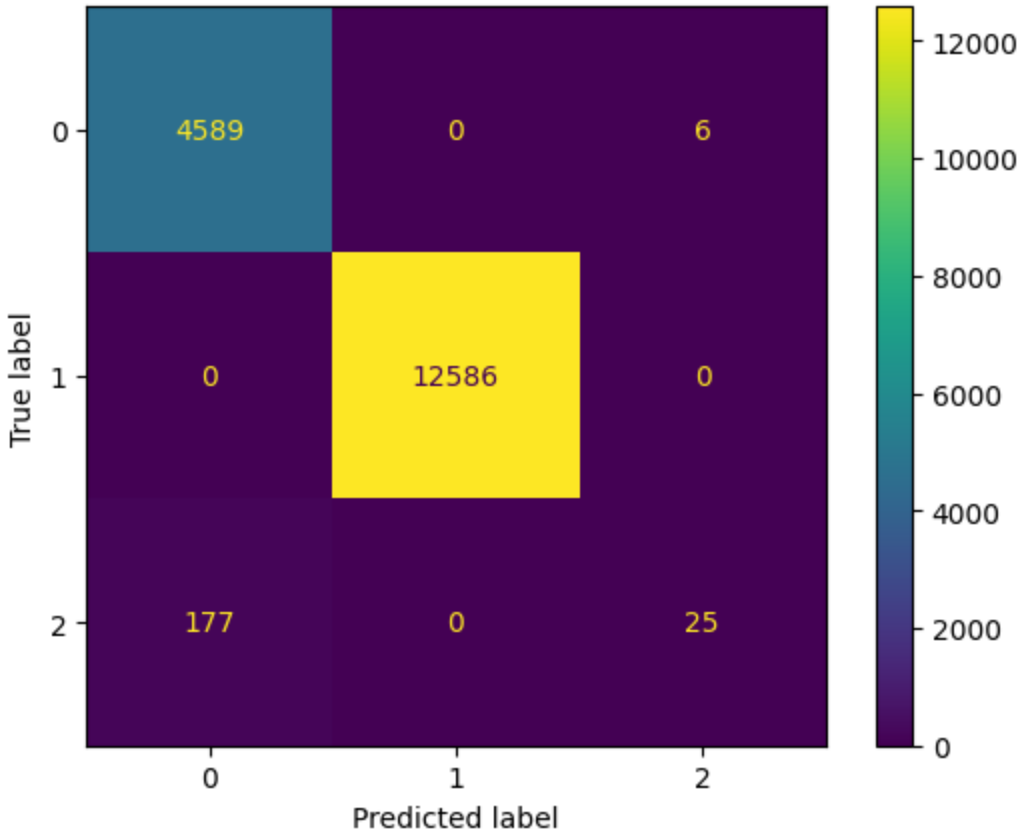
Random Forest

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(classification_report(y_test,y_pred))
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	4595
1	1.00	1.00	1.00	12586
2	0.81	0.12	0.21	202

```
    accuracy                           0.99     17383
   macro avg       0.92      0.71      0.73     17383
weighted avg       0.99      0.99      0.99     17383
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9eb0d831c0>
```



4)COMPARING MODEL SOLUTIONS

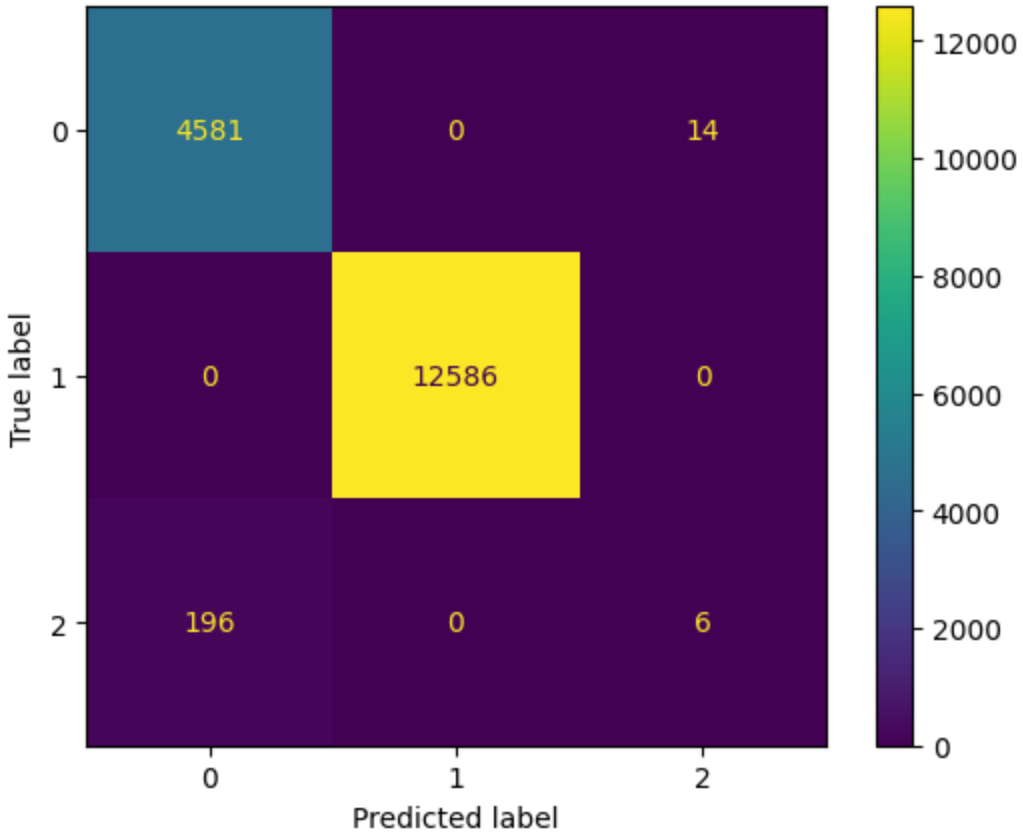
According to the evaluation of the three classification algorithms, random forest gave the best result. The accuracy values of decision tree, logistic regression and random forest are 0.98, 0.78 and 0.99 respectively.

5)Using the tools for increasing the model prediction performance

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X_train_std=scaler.fit_transform(X_train)
X_test_std=scaler.transform(X_test)
clf=LogisticRegression()
clf.fit(X_train_std,y_train)
y_predict=clf.predict(X_test_std)
print(classification_report(y_test,y_predict))
ConfusionMatrixDisplay.from_estimator(clf,X_test_std,y_test)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: 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(
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      4595
```

```
              1       1.00      1.00      1.00     12586
              2       0.30      0.03      0.05       202

    accuracy                            0.99     17383
   macro avg       0.75      0.68      0.68     17383
weighted avg       0.98      0.99      0.98     17383
```



In [ ]:
```python
import numpy as np
from sklearn.model_selection import StratifiedKFold
cv=StratifiedKFold(n_splits=10,shuffle=True)
accuracy_score_train=[]
accuracy_score_test=[]
for train_index, test_index in cv.split(X,y):
  X_train,y_train=X[train_index],y[train_index]
  X_test,y_test=X[test_index],y[test_index]
  scaler=StandardScaler() #mü ve sigma trainden öğrenilip, test setine uygulanır
  X_train_std=scaler.fit_transform(X_train)
  X_test_std=scaler.fit_transform(X_test)
  clf=tree.DecisionTreeClassifier(class_weight='balanced')
  clf.fit(X_train_std,y_train)
  y_predict_test=clf.predict(X_test_std)
  y_predict_train=clf.predict(X_train_std)
  accuracy_score_train.append(accuracy_score(y_train,y_predict_train))
  accuracy_score_test.append(accuracy_score(y_test,y_predict_test))
print("train:",np.mean(accuracy_score_train))
print("test:",np.mean(accuracy_score_test))
```

```
train: 0.9999795455435244
test: 0.9839726477549519
```

In [ ]:
```python
from sklearn.model_selection import cross_val_score
clf = RandomForestClassifier(n_estimators = 100)
cv=StratifiedKFold(n_splits=10,shuffle=True)
scores=cross_val_score(clf,X,y,cv=cv,scoring="accuracy")
print(scores.mean())
```

```
0.9892192231380375
```

In [ ]:
```
from sklearn.pipeline import Pipeline
scaler=StandardScaler()
clf=tree.DecisionTreeClassifier(class_weight='balanced')
pipeline=Pipeline([("scale:",scaler),("estimator:",clf)])
cv=StratifiedKFold(n_splits=10,shuffle=True)
scores=cross_val_score(pipeline,X,y,cv=cv,scoring="accuracy")
print(scores.mean())
```

```
0.9842372602153933
```