In [1]:
```
# Necessary imports

## Data loading, processing and for more
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE

## Visualization
import seaborn as sns
import matplotlib.pyplot as plt
# set seaborn style because it prettier
sns.set()

## Metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve, auc

## Models
import xgboost as xgb
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
```

In [2]:
```
# read the data and show first 5 rows
data = pd.read_csv(r'C:\Users\Tabassum\Desktop\Data mining\bs140513_032310.csv')
data.head(5)
```

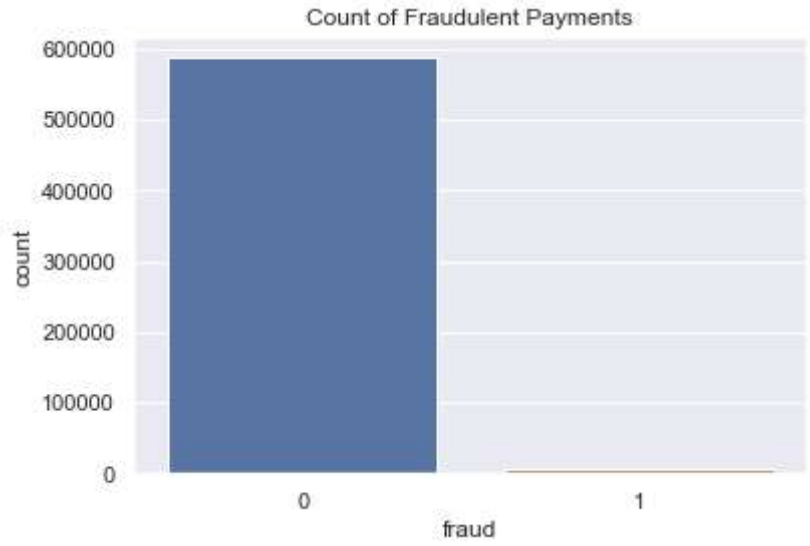
Out[2]:

	step	customer	age	gender	zipcodeOri	merchant	zipMerchant	category	am
0	0	'C1093826151'	'4'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	
1	0	'C352968107'	'2'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	3
2	0	'C2054744914'	'4'	'F'	'28007'	'M1823072687'	'28007'	'es_transportation'	2
3	0	'C1760612790'	'3'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	1
4	0	'C757503768'	'5'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	3

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594643 entries, 0 to 594642
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   step         594643 non-null  int64  
 1   customer     594643 non-null  object 
 2   age          594643 non-null  object 
 3   gender       594643 non-null  object 
 4   zipcodeOri   594643 non-null  object 
 5   merchant     594643 non-null  object 
 6   zipMerchant  594643 non-null  object 
 7   category     594643 non-null  object 
 8   amount       594643 non-null  float64
 9   fraud        594643 non-null  int64  
dtypes: float64(1), int64(2), object(7)
memory usage: 45.4+ MB
```

In [4]:
```
# Create two dataframes with fraud and non-fraud data
df_fraud = data.loc[data.fraud == 1]
df_non_fraud = data.loc[data.fraud == 0]

sns.countplot(x="fraud",data=data)
plt.title("Count of Fraudulent Payments")
plt.show()
print("Number of normal examples: ",df_non_fraud.fraud.count())
print("Number of fradulent examples: ",df_fraud.fraud.count())
#print(data.fraud.value_counts()) # does the same thing above
```




```
Number of normal examples:  587443
Number of fradulent examples:  7200
```

In [5]:
```
print("Mean feature values per category",data.groupby('category')['amount','fraud
```

```
<ipython-input-5-cc3083ece405>:1: FutureWarning: Indexing with multiple keys (i
mplicitly converted to a tuple of keys) will be deprecated, use a list instead.
  print("Mean feature values per category",data.groupby('category')['amount','f
raud'].mean())
```

Mean feature values per category

category	amount	fraud
'es_barsandrestaurants'	43.461014	0.018829
'es_contents'	44.547571	0.000000
'es_fashion'	65.666642	0.017973
'es_food'	37.070405	0.000000
'es_health'	135.621367	0.105126
'es_home'	165.670846	0.152064
'es_hotelservices'	205.614249	0.314220
'es_hyper'	45.970421	0.045917
'es_leisure'	288.911303	0.949900
'es_otherservices'	135.881524	0.250000
'es_sportsandtoys'	215.715280	0.495252
'es_tech'	120.947937	0.066667
'es_transportation'	26.958187	0.000000
'es_travel'	2250.409190	0.793956
'es_wellnessandbeauty'	65.511221	0.047594

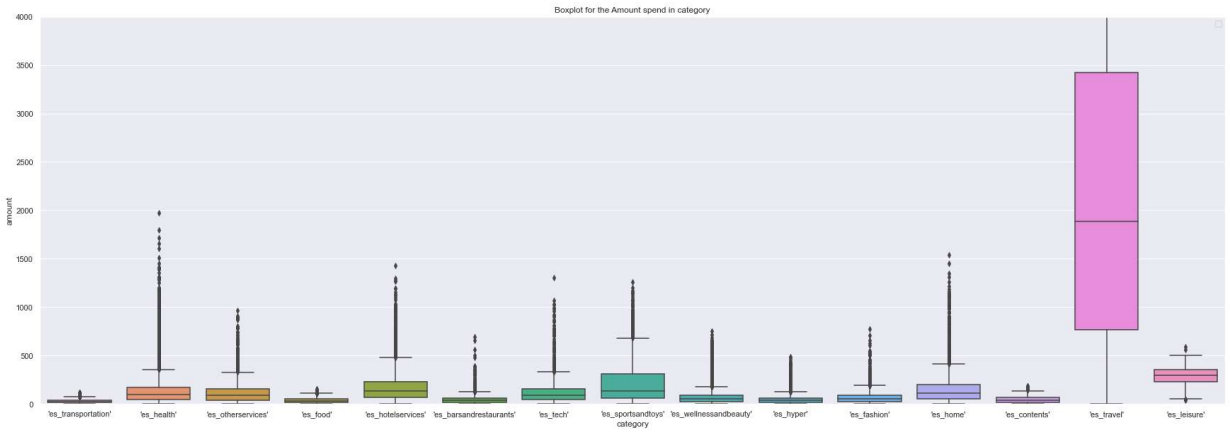
In [6]:
```
# Create two dataframes with fraud and non-fraud data
pd.concat([df_fraud.groupby('category')['amount'].mean(),df_non_fraud.groupby('ca
           data.groupby('category')['fraud'].mean()*100],keys=["Fraudulent","Non-
          sort=False).sort_values(by=['Non-Fraudulent'])
```

Out[6]:

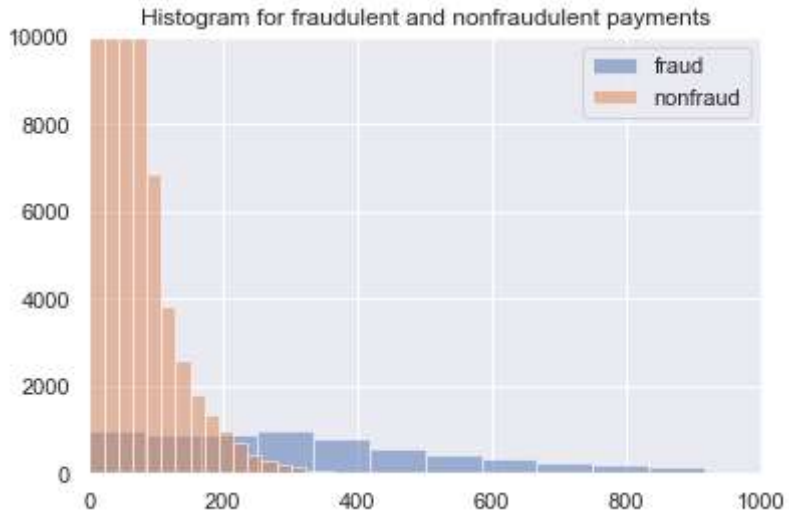
category	Fraudulent	Non-Fraudulent	Percent(%)
'es_transportation'	NaN	26.958187	0.000000
'es_food'	NaN	37.070405	0.000000
'es_hyper'	169.255429	40.037145	4.591669
'es_barsandrestaurants'	164.092667	41.145997	1.882944
'es_contents'	NaN	44.547571	0.000000
'es_wellnessandbeauty'	229.422535	57.320219	4.759380
'es_fashion'	247.008190	62.347674	1.797335
'es_leisure'	300.286878	73.230400	94.989980
'es_otherservices'	316.469605	75.685497	25.000000
'es_sportsandtoys'	345.366811	88.502738	49.525237
'es_tech'	415.274114	99.924638	6.666667
'es_health'	407.031338	103.737228	10.512614
'es_hotelservices'	421.823339	106.548545	31.422018
'es_home'	457.484834	113.338409	15.206445
'es_travel'	2660.802872	669.025533	79.395604

In [7]:
```
# Plot histograms of the amounts in fraud and non-fraud data
plt.figure(figsize=(30,10))
sns.boxplot(x=data.category,y=data.amount)
plt.title("Boxplot for the Amount spend in category")
plt.ylim(0,4000)
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



In [8]:
```
# Plot histograms of the amounts in fraud and non-fraud data
plt.hist(df_fraud.amount, alpha=0.5, label='fraud',bins=100)
plt.hist(df_non_fraud.amount, alpha=0.5, label='nonfraud',bins=100)
plt.title("Histogram for fraudulent and nonfraudulent payments")
plt.ylim(0,10000)
plt.xlim(0,1000)
plt.legend()
plt.show()
```



In [9]:
```
print((data.groupby('age')['fraud'].mean()*100).reset_index().rename(columns={'ag
```

	Age	Fraud Percent
7	'U'	0.594228
6	'6'	0.974826
5	'5'	1.095112
1	'1'	1.185254
3	'3'	1.192815
2	'2'	1.251401
4	'4'	1.293281
0	'0'	1.957586

In [10]:
```
print("Unique zipCodeOri values: ",data.zipcodeOri.nunique())
print("Unique zipMerchant values: ",data.zipMerchant.nunique())
# dropping zipcodeori and zipMerchant since they have only one unique value
data_reduced = data.drop(['zipcodeOri','zipMerchant'],axis=1)
```

```
Unique zipCodeOri values:  1
Unique zipMerchant values:  1
```

In [11]:
```
data_reduced.columns
```

Out[11]:
```
Index(['step', 'customer', 'age', 'gender', 'merchant', 'category', 'amount',
       'fraud'],
      dtype='object')
```

In [12]:
```
# turning object columns type to categorical for easing the transformation proces
col_categorical = data_reduced.select_dtypes(include= ['object']).columns
for col in col_categorical:
    data_reduced[col] = data_reduced[col].astype('category')
# categorical values ==> numeric values
data_reduced[col_categorical] = data_reduced[col_categorical].apply(lambda x: x.c
data_reduced.head(5)
```

Out[12]:

	step	customer	age	gender	merchant	category	amount	fraud
0	0	210	4	2	30	12	4.55	0
1	0	2753	2	2	30	12	39.68	0
2	0	2285	4	1	18	12	26.89	0
3	0	1650	3	2	30	12	17.25	0
4	0	3585	5	2	30	12	35.72	0

In [13]:
```
X = data_reduced.drop(['fraud'],axis=1)
y = data['fraud']
print(X.head(),"\n")
print(y.head())
```

```
   step  customer  age  gender  merchant  category  amount
0     0       210    4       2        30        12    4.55
1     0      2753    2       2        30        12   39.68
2     0      2285    4       1        18        12   26.89
3     0      1650    3       2        30        12   17.25
4     0      3585    5       2        30        12   35.72

0    0
1    0
2    0
3    0
4    0
Name: fraud, dtype: int64
```

In [14]:
```
y[y==1].count()
```

Out[14]: 7200

In [15]:
```
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)
y_res = pd.DataFrame(y_res)
print(y_res.value_counts())
```

```
fraud
0        587443
1        587443
dtype: int64
```

In [16]:
```
# I won't do cross validation since we have a lot of instances
X_train, X_test, y_train, y_test = train_test_split(X_res,y_res,test_size=0.3,ran
```

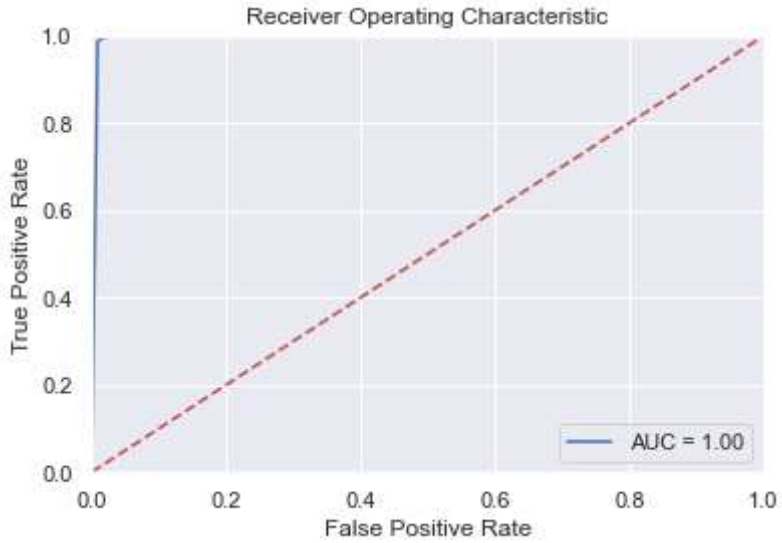
In [17]:
```
# %% Function for plotting ROC_AUC curve

def plot_roc_auc(y_test, preds):
    '''
    Takes actual and predicted(probabilities) as input and plots the Receiver
    Operating Characteristic (ROC) curve
    '''
    fpr, tpr, threshold = roc_curve(y_test, preds)
    roc_auc = auc(fpr, tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0, 1])
    plt.ylim([0, 1])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

In [18]:
```
# The base score should be better than predicting always non-fraduelent
print("Base accuracy score we must beat is: ",
      df_non_fraud.fraud.count()/ np.add(df_non_fraud.fraud.count(),df_fraud.frau
```

```
Base accuracy score we must beat is:  98.7891894800746
```

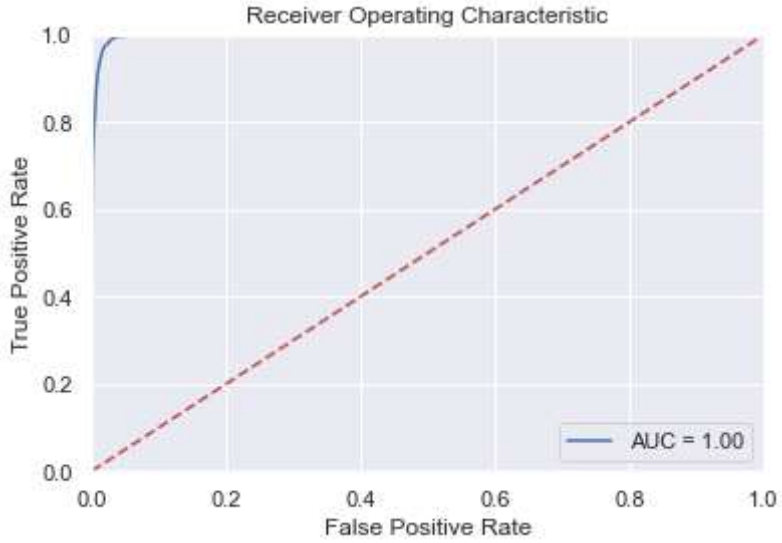
In [19]:
```
# %% K-ello Neigbors

knn = KNeighborsClassifier(n_neighbors=5,p=1)

knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)


print("Classification Report for K-Nearest Neighbours: \n", classification_report
print("Confusion Matrix of K-Nearest Neigbours: \n", confusion_matrix(y_test,y_pr
plot_roc_auc(y_test, knn.predict_proba(X_test)[:,1])
```

```
C:\Users\Tabassum\anaconda3\lib\site-packages\sklearn\neighbors\_classificatio
n.py:179: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples,), for example using ra
vel().
  return self._fit(X, y)

Classification Report for K-Nearest Neighbours:
               precision    recall  f1-score   support

           0       1.00      0.98      0.99    176233
           1       0.98      1.00      0.99    176233

    accuracy                           0.99    352466
   macro avg       0.99      0.99      0.99    352466
weighted avg       0.99      0.99      0.99    352466

Confusion Matrix of K-Nearest Neigbours:
 [[171999   4234]
 [   362 175871]]
```



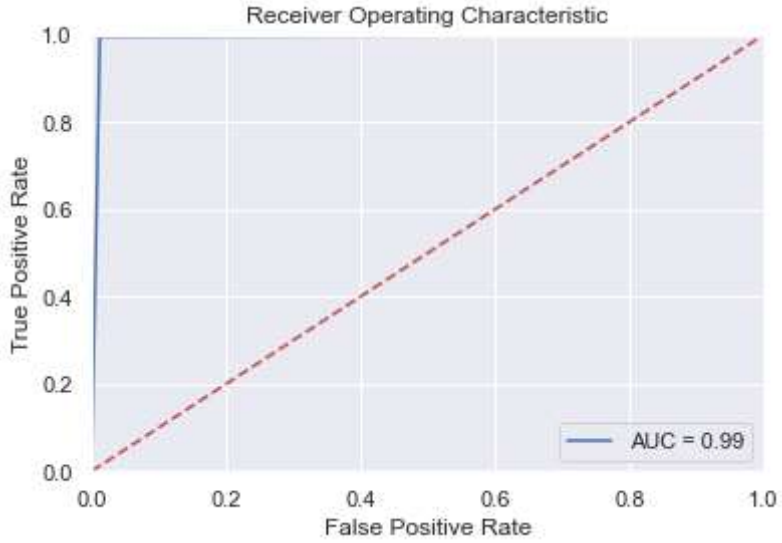
In [20]:
```
# %% Random Forest Classifier

rf_clf = RandomForestClassifier(n_estimators=100,max_depth=8,random_state=42,
                                verbose=1,class_weight="balanced")

rf_clf.fit(X_train,y_train)
y_pred = rf_clf.predict(X_test)

print("Classification Report for Random Forest Classifier: \n", classification_re
print("Confusion Matrix of Random Forest Classifier: \n", confusion_matrix(y_test
plot_roc_auc(y_test, rf_clf.predict_proba(X_test)[:,1])
```

```
<ipython-input-20-e57e8f380b47>:6: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_sample
s,), for example using ravel().
  rf_clf.fit(X_train,y_train)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:  4.5min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    7.1s finished

Classification Report for Random Forest Classifier:
               precision    recall  f1-score   support

           0       0.99      0.97      0.98    176233
           1       0.97      0.99      0.98    176233

    accuracy                           0.98    352466
   macro avg       0.98      0.98      0.98    352466
weighted avg       0.98      0.98      0.98    352466

Confusion Matrix of Random Forest Classifier:
 [[170106   6127]
 [  1079 175154]]

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worker
s.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    7.1s finished
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 1.00
0.0
0.2
0.4
0.6
0.8
1.0

In [22]:
```
XGBoost_CLF = xgb.XGBClassifier(max_depth=6, learning_rate=0.05, n_estimators=400
                                objective="binary:hinge", booster='gbtree',
                                n_jobs=-1, nthread=None, gamma=0, min_child_weigh
                                subsample=1, colsample_bytree=1, colsample_byleve
                                scale_pos_weight=1, base_score=0.5, random_state=

XGBoost_CLF.fit(X_train,y_train)

y_pred = XGBoost_CLF.predict(X_test)

print("Classification Report for XGBoost: \n", classification_report(y_test, y_pr
print("Confusion Matrix of XGBoost: \n", confusion_matrix(y_test,y_pred))
plot_roc_auc(y_test, XGBoost_CLF.predict_proba(X_test)[:,1])
```

```
C:\Users\Tabassum\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarn
ing: The use of label encoder in XGBClassifier is deprecated and will be remove
d in a future release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode y
our labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\Tabassum\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: D
ataConversionWarning: A column-vector y was passed when a 1d array was expecte
d. Please change the shape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)

[11:31:54] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:573:
Parameters: { "scale_pos_weight", "verbose" } might not be used.

  This may not be accurate due to some parameters are only used in language bin
dings but
  passed down to XGBoost core.  Or some parameters are not used but slip throug
h this
  verification. Please open an issue if you find above cases.


Classification Report for XGBoost:
               precision    recall  f1-score   support

           0       1.00      0.99      0.99    176233
           1       0.99      1.00      0.99    176233

    accuracy                           0.99    352466
   macro avg       0.99      0.99      0.99    352466
weighted avg       0.99      0.99      0.99    352466

Confusion Matrix of XGBoost:
 [[174047   2186]
 [   706 175527]]
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
AUC = 0.99
0.0
0.2
0.4
0.6
0.8
1.0