In [5]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [6]:
```
df=pd.read_csv("creditcard.csv")
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
Time      284807 non-null float64
V1        284807 non-null float64
V2        284807 non-null float64
V3        284807 non-null float64
V4        284807 non-null float64
V5        284807 non-null float64
V6        284807 non-null float64
V7        284807 non-null float64
V8        284807 non-null float64
V9        284807 non-null float64
V10       284807 non-null float64
V11       284807 non-null float64
V12       284807 non-null float64
V13       284807 non-null float64
V14       284807 non-null float64
V15       284807 non-null float64
V16       284807 non-null float64
V17       284807 non-null float64
V18       284807 non-null float64
V19       284807 non-null float64
V20       284807 non-null float64
V21       284807 non-null float64
V22       284807 non-null float64
V23       284807 non-null float64
V24       284807 non-null float64
V25       284807 non-null float64
V26       284807 non-null float64
V27       284807 non-null float64
V28       284807 non-null float64
Amount    284807 non-null float64
Class     284807 non-null int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [8]:
```
df.head()
```

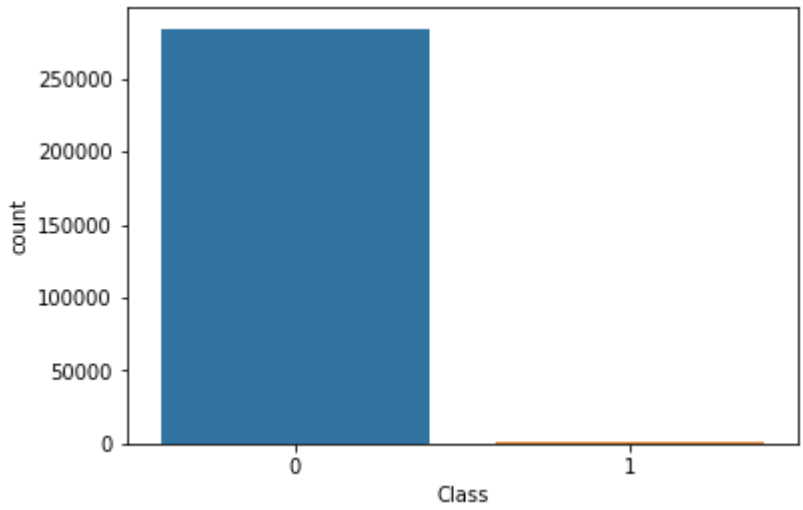
Out[8]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

In [9]:
```
sns.countplot(x='Class',data=df)
```

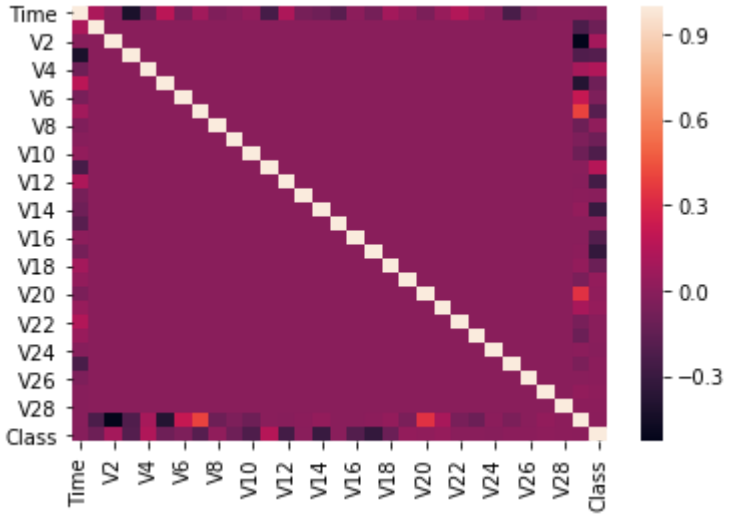
Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29db574a668>
```



In [10]:
```
#sns.pairplot(df.sample(1000))
```

In [11]:
```
sns.heatmap(df.corr())
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x29db63553c8>



In [12]:
```
df.drop('Time',axis=1,inplace=True)
```

In [13]:
```
df.head(1)
```

Out[13]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787

1 rows × 30 columns

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
X=df.drop('Class',axis=1)
y=df['Class']
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ran
```

In [17]:
```
from sklearn.linear_model import LinearRegression
```

In [18]:
```
lm = LinearRegression()
```

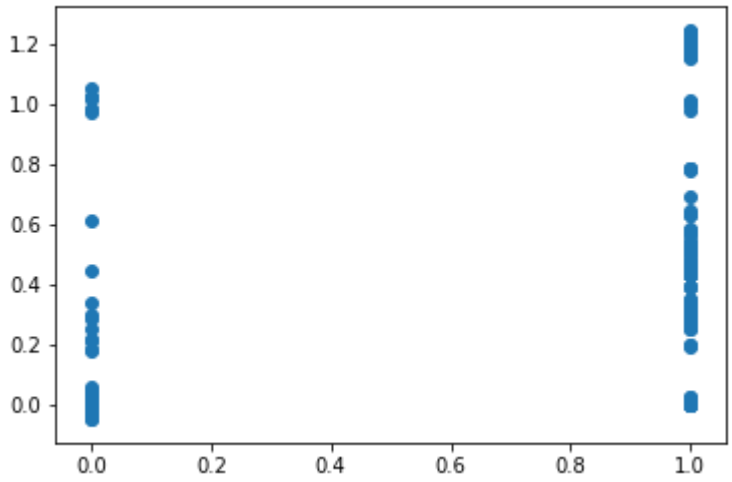
In [19]:
```
lm.fit(X_train,y_train)
```

Out[19]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fa
lse)
```

In [20]:
```
predictions = lm.predict(X_test)
```

In [21]:
```
plt.scatter(y_test,predictions)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x29db68c76d8>
```



In [22]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
```

In [23]:
```
print(classification_report(y_test,predictions))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-23-49b4b26ceb45> in <module>()
----> 1 print(classification_report(y_test,predictions))

F:\anaconda\lib\site-packages\sklearn\metrics\_classification.py in classif
ication_report(y_true, y_pred, labels, target_names, sample_weight, digits,
output_dict, zero_division)
   1969     """
   1970
-> 1971     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   1972
   1973     labels_given = True

F:\anaconda\lib\site-packages\sklearn\metrics\_classification.py in _check_
targets(y_true, y_pred)
     88     if len(y_type) > 1:
     89         raise ValueError("Classification metrics can't handle a mix
of {0} "
---> 90                          "and {1} targets".format(type_true, type_p
red))
     91
     92     # We can't have more than one value on y_type => The set is no
 more needed

ValueError: Classification metrics can't handle a mix of binary and continu
ous targets
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
```

In [25]:
```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
F:\anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:940: Conver
genceWarning: 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 (https://sci
kit-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-regr
ession (https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[25]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [26]:
```
predictions = logmodel.predict(X_test)
```

In [27]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71077
           1       0.88      0.61      0.72       125

    accuracy                           1.00     71202
   macro avg       0.94      0.80      0.86     71202
weighted avg       1.00      1.00      1.00     71202
```

In [28]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[71067    10]
 [   49    76]]
```

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [30]:
```
knn = KNeighborsClassifier(n_neighbors=1)
```

In [31]:
```
knn.fit(X_train,y_train)
```

Out[31]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [32]:
```
pred = knn.predict(X_test)
```

In [33]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71077
           1       0.88      0.73      0.80       125

    accuracy                           1.00     71202
   macro avg       0.94      0.86      0.90     71202
weighted avg       1.00      1.00      1.00     71202
```

In [34]:
```
print(confusion_matrix(y_test,pred))
```

```
[[71065    12]
 [   34    91]]
```

In [ ]:
```
error_rate = []

# Will take some time
for i in range(1,10):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [35]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [36]:
```
dtree = DecisionTreeClassifier()
```

In [38]:
```
dtree.fit(X_train,y_train)
```

Out[38]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=No
ne,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [39]:
```
predictions = dtree.predict(X_test)
```

In [40]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71077
           1       0.72      0.78      0.75       125

    accuracy                           1.00     71202
   macro avg       0.86      0.89      0.88     71202
weighted avg       1.00      1.00      1.00     71202
```

In [41]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[71039    38]
 [   27    98]]
```

In [42]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

Out[42]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='aut
o',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [43]:
```
rfc_pred = rfc.predict(X_test)
```

In [44]:
```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[71070     7]
 [   25   100]]
```

In [45]:
```
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     71077
           1       0.93      0.80      0.86       125

    accuracy                           1.00     71202
   macro avg       0.97      0.90      0.93     71202
weighted avg       1.00      1.00      1.00     71202
```

In [ ]:
```

```