In [1]:

```
import pandas as pd

payments = pd.read_csv('./csv/olist_order_payments_dataset.csv')
payments.head()
```

Out[1]:

	order_id	payment_sequential	payment_type	payment_installments
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2

Payment types

In [2]:

```
payments_type = payments.groupby(by="payment_type").agg({'payment_value': 'mean', 'order_ic
payments_type
```

Out[2]:

	payment_value	order_id
payment_type		
boleto	145.034435	19784
credit_card	163.319021	76795
debit_card	142.570170	1529
not_defined	0.000000	3
voucher	65.703354	5775

Count of sales

In [3]:

```
import seaborn as sns

payments_type.sort_values(by="order_id", ascending=False, inplace=True)
sns.barplot(payments_type.index, payments_type['order_id'])
```

Out[3]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2e1dd4ffbe0>
```

In [4]:

```
sales = pd.read_csv('./csv/olist_orders_dataset.csv')
sales.head()
```

Out[4]:

	order_id	customer_id	order_status	order_p
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	

In [5]:

```
sales = pd.merge(sales, payments, how='inner')
sales.head()
```

Out[5]:

	order_id	customer_id	order_status	order_p
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	

In [6]:

```
dummies = sales.payment_type.str.get_dummies()
dummies.head()
```

Out[6]:

	boleto	credit_card	debit_card	not_defined	voucher
0	0	1	0	0	0
1	0	0	0	0	1
2	0	0	0	0	1
3	1	0	0	0	0
4	0	1	0	0	0

In [7]:

```
sales_payment = pd.concat([sales, dummies], axis=1)
sales_payment.head()
```

Out[7]:

	order_id	customer_id	order_status	order_p
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	

Time series projetion

In [8]:

```
sales_payment['date'] = pd.to_datetime(sales_payment['order_approved_at'])
sales_payment.head()
```

Out[8]:

	order_id	customer_id	order_status	order_p
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	

Group by period

In [9]:

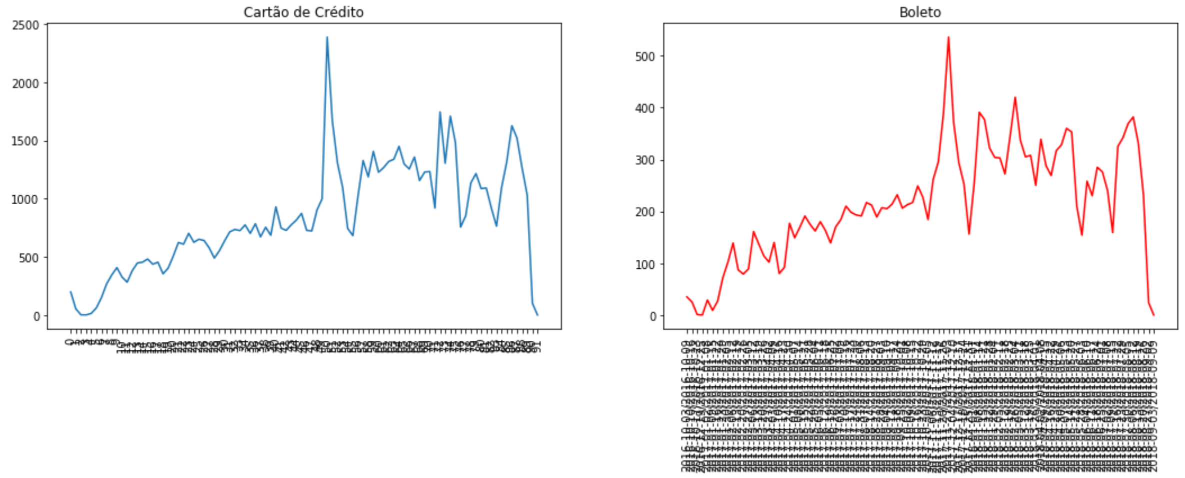
```
mensal = sales_payment.groupby(by=sales_payment.date.dt.to_period("W")).agg({'credit_card':
mensal.rename(columns={'order_id': 'count'}, inplace=True)
mensal.reset_index(inplace=True)
mensal.head()
```

Out[9]:

	date	credit_card	boleto	voucher	count
0	2016-10-03/2016-10-09	198	35	14	249
1	2016-10-10/2016-10-16	53	25	8	86
2	2016-10-17/2016-10-23	2	1	0	3
3	2016-12-19/2016-12-25	1	0	0	1
4	2017-01-02/2017-01-08	14	29	4	47

In [10]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, mensal.shape[0], 1)

#                       X, Y
fig, ax = plt.subplots(1, 2, figsize=(18,5))

ax[0].set_title('Cartão de Crédito')
ax[0].plot(x, mensal.credit_card)
ax[0].xaxis.set_tick_params(rotation=90)
ax[0].set_xticks(x)
# ax[0].set_xticklabels(mensal['date'])

ax[1].set_title('Boleto')
ax[1].plot(x, mensal.boleto, color = 'r')
ax[1].xaxis.set_tick_params(rotation=90)
ax[1].set_xticks(x)
ax[1].set_xticklabels(mensal['date'])

plt.show()
```



In [11]:

```
import datetime

dates = pd.Series([datetime.datetime(period.year, period.month, 1) for period in mensal['da
mensal['date'] = dates
mensal.head()
```

Out[11]:

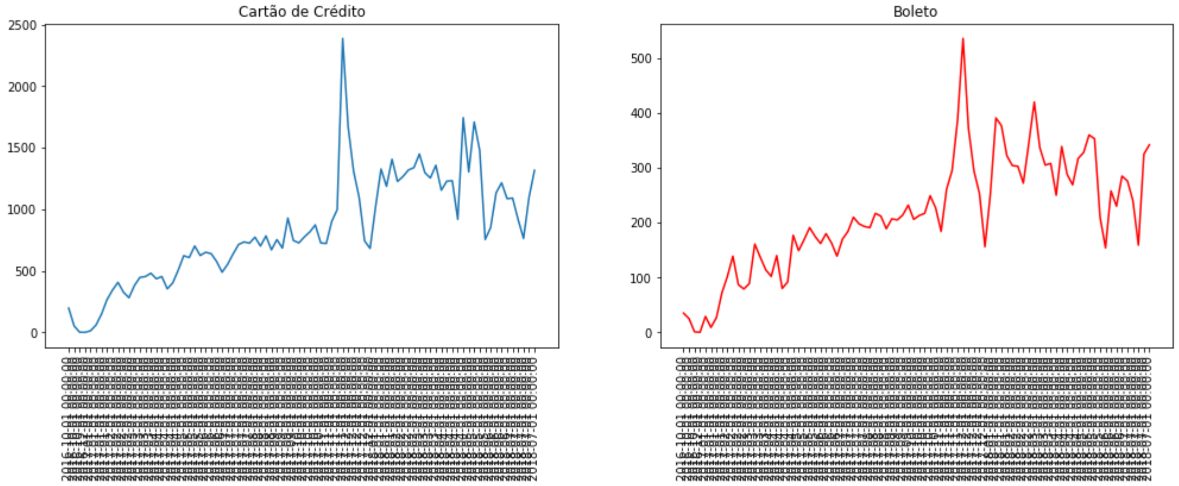
	date	credit_card	boleto	voucher	count
0	2016-10-01	198	35	14	249
1	2016-10-01	53	25	8	86
2	2016-10-01	2	1	0	3
3	2016-12-01	1	0	0	1
4	2017-01-01	14	29	4	47

In [12]:

```
mask = (mensal['date'] > datetime.datetime(2016, 9, 1)) & (mensal['date'] <= datetime.datet
mensal = mensal[mask]
mensal.reset_index(inplace=True)
```

In [13]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, mensal.shape[0], 1)

#                      X, Y
fig, ax = plt.subplots(1, 2, figsize=(17,5))

ax[0].set_title('Cartão de Crédito')
ax[0].plot(x, mensal.credit_card)
ax[0].xaxis.set_tick_params(rotation=90)
ax[0].set_xticks(mensal.index)
ax[0].set_xticklabels(mensal['date'])

ax[1].set_title('Boleto')
ax[1].plot(x, mensal.boleto, color = 'r')
ax[1].xaxis.set_tick_params(rotation=90)
ax[1].set_xticks(x)
ax[1].set_xticklabels(mensal['date'])

plt.show()
```



Base line prediction using Support Vector Regression

In [14]:

```
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler

x = np.arange(0, mensal.shape[0], 1)
x = [[v] for v in x]

credit_card = [[x] for x in list(mensal.credit_card)]
credit_card

scaler = MinMaxScaler().fit(credit_card)
mensal['credit_card_p'] = pd.Series([x[0] for x in scaler.transform(credit_card)])

svm = SVR(coef0=0.1)
svm.fit(x, mensal.credit_card_p)

net = MLPRegressor()
net.fit(x, mensal.credit_card_p)

mensal.credit_card_p
```

```
12     0.159262
13     0.186924
14     0.189858
15     0.201174
16     0.182313
17     0.189858
18     0.147946
19     0.168483
20     0.212070
21     0.260687
22     0.254401
23     0.293797
24     0.261526
25     0.272422
26     0.267812
27     0.240989
28     0.204526
29     0.231350
         ...
56     0.426655
```

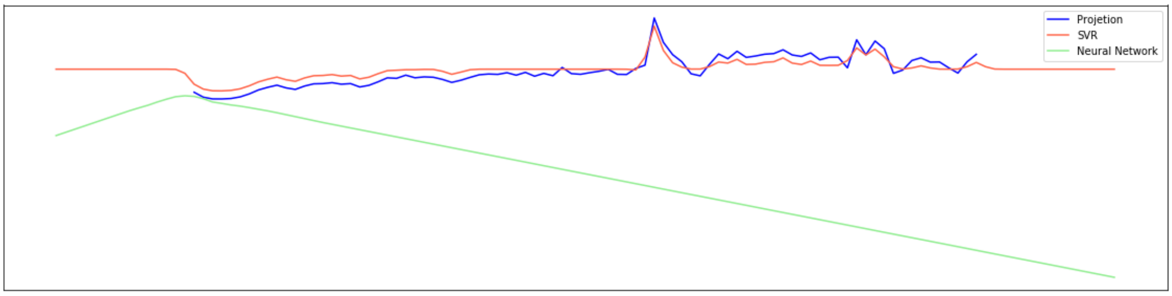
In [15]:

```
x = list(range(-15, mensal.shape[0] + 15))
x = np.array([[v] for v in x])

svr_result = pd.Series(svm.predict(x))
net_result = pd.Series(net.predict(x))
```

In [16]:

```
import matplotlib.pyplot as plt
import numpy as np

x1 = np.arange(x.min(), x.max()+1, 1)
print(len(svr_result), len(x1))

x = np.arange(0, mensal.shape[0], 1)

plt.figure(figsize=(20, 5))

plt.plot(x, mensal.credit_card_p, color = 'blue')
plt.plot(x1, svr_result, color = 'tomato')
plt.plot(x1, net_result, color = 'lightgreen')

plt.xticks([])
plt.yticks([])

plt.legend(['Projetion', 'SVR', 'Neural Network'])

plt.show()
```

```
116 116
```



Base line prediction using LSTM Tensorflow Keras

In [17]:

```
!pip install -q tensorflow keras
```

In [18]:

```
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from keras import optimizers


x = np.arange(0, mensal.shape[0], 1)
x = np.array([[v] for v in x])
x = x.reshape((x.shape[0], 1, x.shape[1]))

y = mensal.credit_card_p


# design network
model = Sequential()

model.add(LSTM(50, return_sequences=True, input_shape = (x.shape[1], x.shape[2]), kernel_ir
model.add(Dropout(0.2))


model.add(LSTM(150, return_sequences=True))
model.add(Dropout(0.2))


model.add(LSTM(150))
model.add(Dropout(0.2))

# model.add(Dense(20,activation='relu'))
model.add(Dense(1,activation='sigmoid'))

# compile
optimizer = optimizers.RMSprop(lr=1e-3)
model.compile(loss='mean_squared_error', optimizer=optimizer)

# fit network
history = model.fit(x, y, epochs=300, batch_size=1000, validation_split=0.1, verbose=1, shu
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\sit
e-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second ar
gument of issubdtype from `float` to `np.floating` is deprecated. In futur
e, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr.
W0709 11:09:07.114496 10816 deprecation_wrapper.py:119] From C:\Program Fi
les (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\ke
ras\backend\tensorflow_backend.py:74: The name tf.get_default_graph is dep
recated. Please use tf.compat.v1.get_default_graph instead.

W0709 11:09:07.127462 10816 deprecation_wrapper.py:119] From C:\Program Fi
les (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\ke
ras\backend\tensorflow_backend.py:517: The name tf.placeholder is deprecat
ed. Please use tf.compat.v1.placeholder instead.

W0709 11:09:07.129457 10816 deprecation_wrapper.py:119] From C:\Program Fi
les (x86)\Microsoft Visual Studio\Shared\Anaconda3_64\lib\site-packages\ke
```

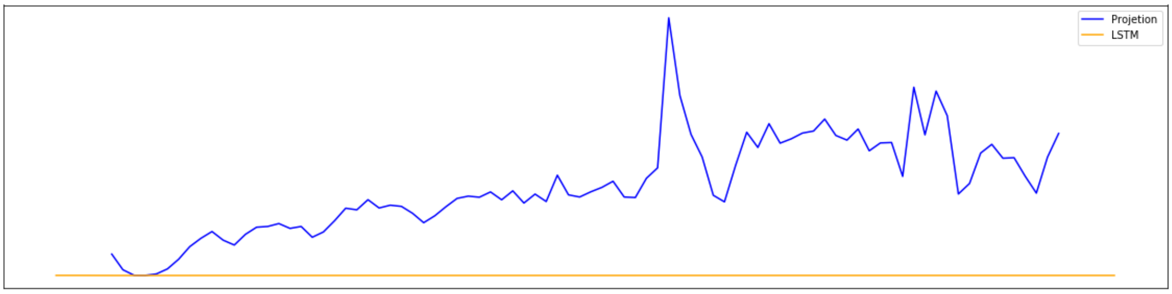
In [19]:

```
x_f = list(range(-5, mensal.shape[0] + 5))
x1 = np.array([[v] for v in x_f])
print(x1.shape)
x1 = x1.reshape((x1.shape[0], 1, x1.shape[1]))

result = pd.Series([v[0] for v in model.predict(x1)])
```

```
(96, 1)
```

In [20]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, mensal.shape[0], 1)

plt.figure(figsize=(20, 5))

plt.plot(x, mensal.credit_card, color = 'blue')
plt.plot(x_f, result, color = 'orange')

plt.xticks([])
plt.yticks([])

plt.legend(['Projetion', 'LSTM'])

plt.show()
```



ARIMA

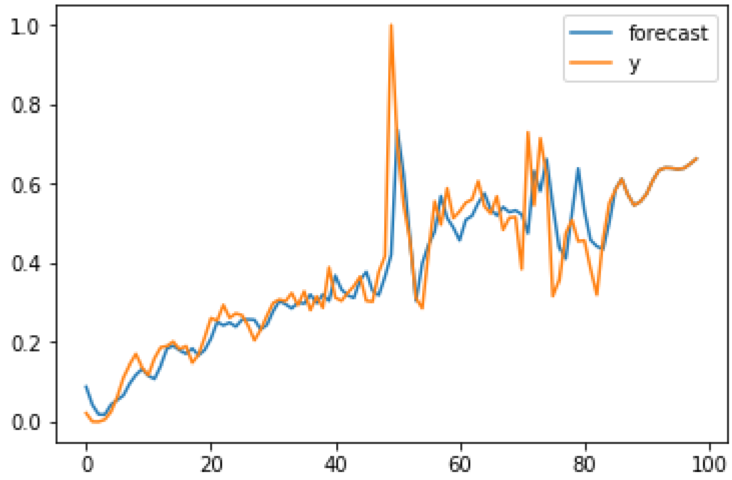
In [21]:

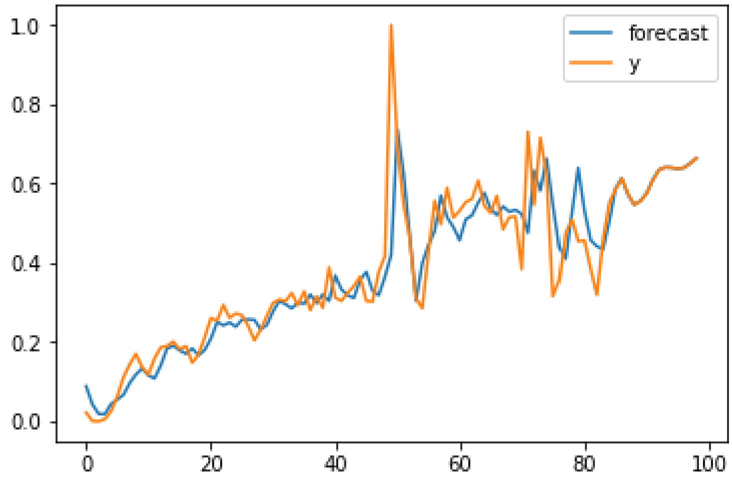
```
!pip install -q statsmodels
```

In [22]:

```
from statsmodels.tsa.arima_model import ARIMA

predictions = list()
actual = [x for x in mensal.credit_card_p]

predictions.append(actual[-1])

for _ in range(15):
    model = ARIMA(actual, order=(6, 1, 1))
    model_fit = model.fit(disp=0)
    predictions.append(model_fit.forecast()[0][0])
    actual.append(model_fit.forecast()[0][0])

model_fit.plot_predict()
```

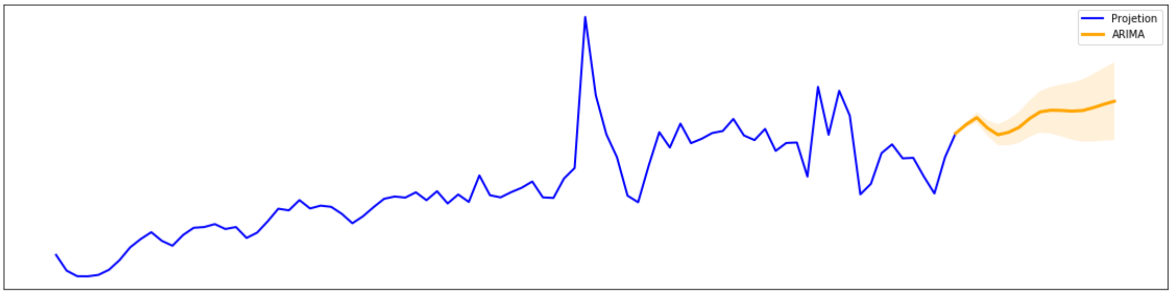
Out[22]:





In [23]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, mensal.shape[0], 1)
x_f = np.arange(mensal.shape[0]-1, mensal.shape[0] - 1 + len(predictions), 1)
sigma = np.arange(0, len(predictions), 1)
sigma = sigma * 0.01

plt.figure(figsize=(20, 5))

plt.plot(x, mensal.credit_card_p, color = 'blue', linewidth=2)
plt.plot(x_f, predictions, color = 'orange', linewidth=3)
plt.fill_between(x_f, predictions+sigma, predictions-sigma, facecolor='orange', alpha=0.15)

plt.xticks([])
plt.yticks([])

plt.legend(['Projetion', 'ARIMA'])

plt.show()
```



In [ ]: