Customer Lifetime Value (CLV) basics

MACHINE LEARNING FOR MARKETING IN PYTHON

Karolis Urbonas
Head of Analytics & Science, Amazon

What is CLV?

- Measurement of customer value
- Can be historical or predicted
- Multiple approaches, depends on business type
- Some methods are formula-based, some are predictive and distribution based

Historical CLV

- Sum revenue of all past transactions
- Multiply by the profit margin
- Alternatively - sum profit of all past transactions, if available
- **Challenge 1** - does not account for tenure, retention and churn
- **Challenge 2** - does not account for new customers and their future revenue

$$\text{Historcal CLV} = (\text{Revenue 1} + \text{Revenue 2} + \ \ldots + \text{Revenue N}) * \text{Profit Margin}$$

Basic CLV formula

- Multiply average revenue with profit margin to get average profit
- Multiply it with average customer lifespan

$$\text{CLV} = \text{Average Revenue} * \text{Profit Margin} * \text{Average Lifespan}$$

Granular CLV formula

- Multiply average revenue per purchase with average frequency and with profit margin
- Multiply it with average customer lifespan
- Accounts for both average revenue per transaction and average frequency per period

$$\text{CLV} = (\text{Avg. revenue per purchase} * \text{Avg. Frequency} * \text{Profit Margin}) * \text{Avg. Lifespan}$$

Traditional CLV formula

- Multiply average revenue with profit margin
- Multiple average profit with the retention to churn rate
- Churn can be derived from retention and equals 1 minus retention rate
- Accounts for customer loyalty, most popular approach

$$\text{CLV} = (\text{Average Revenue} * \text{Profit Margin}) * \frac{\text{Retention Rate}}{\text{Churn Rate}}$$

Introduction to transactions dataset

- Online retail dataset
- Transactions with spent, quantity and other values

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSum	InvoiceMonth
416792	572558	22745	POPPY'S PLAYHOUSE BEDROOM	6	2011-10-25 08:26:00	2.10	14286.0	United Kingdom	12.60	2011-10
482904	577485	23196	VINTAGE LEAF MAGNETIC NOTEPAD	1	2011-11-20 11:56:00	1.45	16360.0	United Kingdom	1.45	2011-11
263743	560034	23299	FOOD COVER WITH BEADS SET 2	6	2011-07-14 13:35:00	3.75	13933.0	United Kingdom	22.50	2011-07
495549	578307	72349B	SET/6 PURPLE BUTTERFLY T-LIGHTS	1	2011-11-23 15:53:00	2.10	17290.0	United Kingdom	2.10	2011-11
204384	554656	21756	BATH BUILDING BLOCK WORD	3	2011-05-25 13:36:00	5.95	17663.0	United Kingdom	17.85	2011-05

Introduction to cohorts dataset

- Derived from online retail dataset
- Assigned acquisition month
- Pivot table with customer counts in subsequent months after acquisition
- Will use it to calculate retention rate

CohortIndex	1	2	3	4	5	6	7	8	9	10	11	12	13
AcquisitionMonth													
2010-12	716.0	246.0	221.0	251.0	245.0	285.0	249.0	236.0	240.0	265.0	254.0	348.0	172.0
2011-01	332.0	69.0	82.0	81.0	110.0	90.0	82.0	86.0	104.0	102.0	124.0	45.0	NaN
2011-02	316.0	58.0	57.0	83.0	85.0	74.0	80.0	83.0	86.0	95.0	28.0	NaN	NaN
2011-03	388.0	63.0	100.0	76.0	83.0	67.0	98.0	85.0	107.0	38.0	NaN	NaN	NaN
2011-04	255.0	49.0	52.0	49.0	47.0	52.0	56.0	59.0	17.0	NaN	NaN	NaN	NaN
2011-05	249.0	40.0	43.0	36.0	52.0	58.0	61.0	22.0	NaN	NaN	NaN	NaN	NaN
2011-06	207.0	33.0	26.0	41.0	49.0	62.0	19.0	NaN	NaN	NaN	NaN	NaN	NaN
2011-07	173.0	28.0	31.0	38.0	44.0	17.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-08	139.0	30.0	28.0	35.0	14.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-09	279.0	56.0	78.0	34.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2011-10	[illegible]	67.0	30.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Calculate monthly retention

Use first month values to calculate cohort sizes

```
cohort_sizes = cohort_counts.iloc[:,0]
```

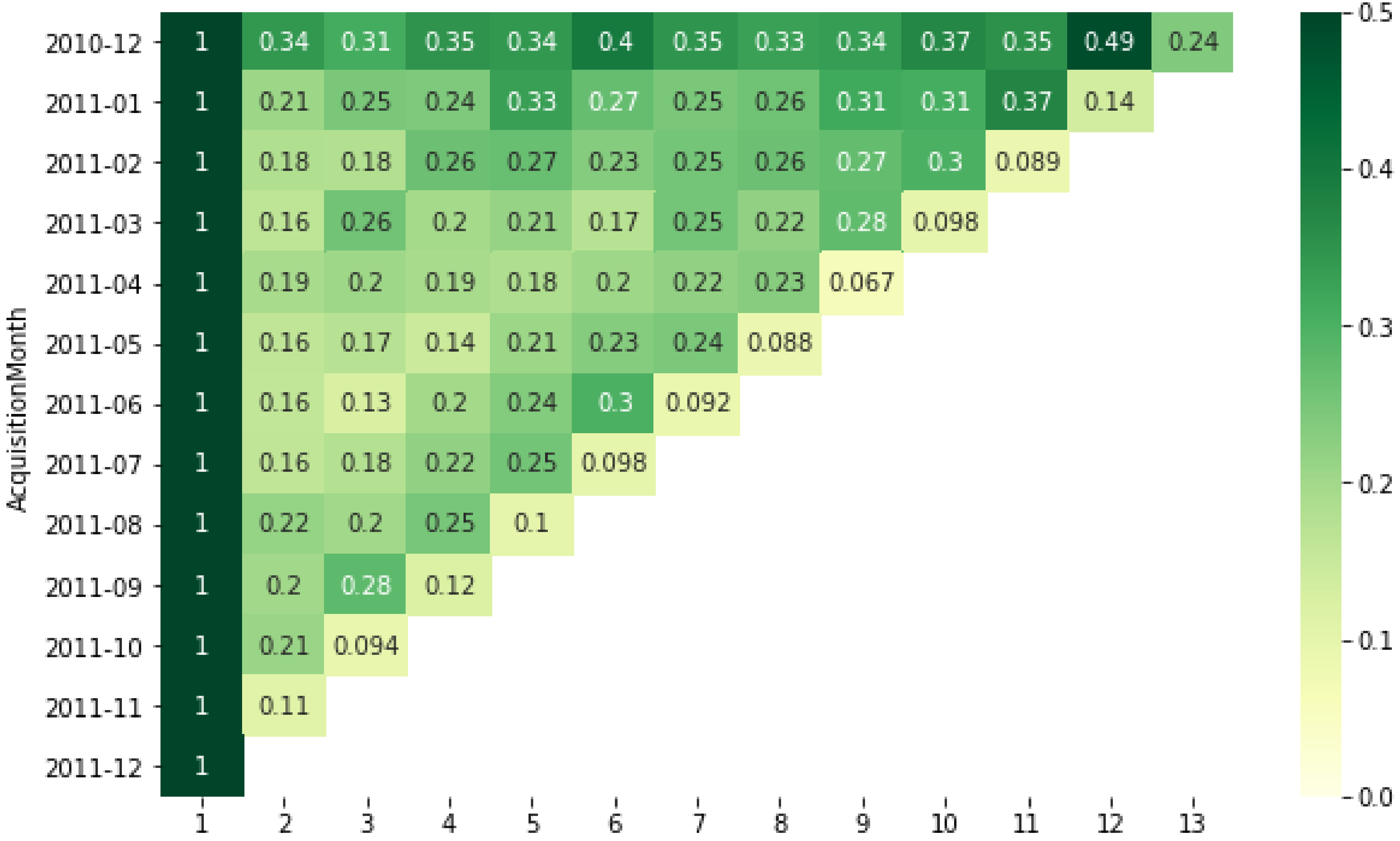
Calculate retention by dividing monthly active users by their initial sizes and derive churn values

```
retention = cohort_counts.divide(cohort_sizes, axis=0)
churn = 1 - retention
```

Plot the retention values in a heatmap

```
sns.heatmap(retention, annot=True, vmin=0, vmax=0.5, cmap="YlGn")
```

Retention table



Let's calculate some CLV metrics!

MACHINE LEARNING FOR MARKETING IN PYTHON

Calculating and projecting CLV

MACHINE LEARNING FOR MARKETING IN PYTHON

Karolis Urbonas
Head of Analytics & Science, Amazon

The goal of CLV

- Measure customer value in revenue / profit
- Benchmark customers
- Identify maximum investment into customer acquisition
- In our case - we'll skip the profit margin for simplicity and use revenue-based CLV formulas

$$\text{CLV} = \text{Average Revenue} * \frac{\text{Retention Rate}}{\text{Churn Rate}}$$

Basic CLV calculation

```
# Calculate monthly spend per customer
monthly_revenue = online.groupby(['CustomerID','InvoiceMonth'])['TotalSum'].sum().mean()
# Calculate average monthly spend
monthly_revenue = np.mean(monthly_revenue)
# Define lifespan to 36 months
lifespan_months = 36
# Calculate basic CLV
clv_basic = monthly_revenue * lifespan_months
# Print basic CLV value
print('Average basic CLV is {:.1f} USD'.format(clv_basic))
```

```
Average basic CLV is 4774.6 USD
```

Granular CLV calculation

```
# Calculate average revenue per invoice
revenue_per_purchase = online.groupby(['InvoiceNo'])['TotalSum'].mean().mean()
# Calculate average number of unique invoices per customer per month
freq = online.groupby(['CustomerID','InvoiceMonth'])['InvoiceNo'].nunique().mean()
# Define lifespan to 36 months
lifespan_months = 36
# Calculate granular CLV
clv_granular = revenue_per_purchase * freq * lifespan_months
# Print granular CLV value
print('Average granular CLV is {:.1f} USD'.format(clv_granular))
```

```
Average granular CLV is 1635.2 USD
Revenue per purchase: 34.8 USD
Frequency per month: 1.3
```

Traditional CLV calculation

```
# Calculate monthly spend per customer
monthly_revenue = online.groupby(['CustomerID','InvoiceMonth'])['TotalSum'].sum().mean()
# Calculate average monthly retention rate
retention_rate = retention_rate = retention.iloc[:,1:].mean().mean()
# Calculate average monthly churn rate
churn_rate = 1 - retention_rate
# Calculate traditional CLV
clv_traditional = monthly_revenue * (retention_rate / churn_rate)
# Print traditional CLV and the retention rate values
print('Average traditional CLV is {:.1f} USD at {:.1f} % retention_rate'.format(
                    clv_traditional, retention_rate*100))
```

```
Average traditional CLV is 49.9 USD at 27.3 % retention_rate
Monthly average revenue: 132.6 USD
```

Which method to use?

- Depends on the business model.
- Traditional CLV model - assumes churn is definitive = customer "dies".
- Traditional model is not robust at low retention values - will under-report the CLV.
- Hardest thing to predict - frequency in the future.

Let's calculate customer lifetimes values!

MACHINE LEARNING FOR MARKETING IN PYTHON

Data preparation for purchase prediction

MACHINE LEARNING FOR MARKETING IN PYTHON

Karolis Urbonas
Head of Analytics & Science, Amazon

Regression - predicting continuous variable

- Regression - type of supervised learning
- Target variable - continuous or count variable
- Simplest version - linear regression
- Count data (e.g. number of days active) sometimes better predicted by Poisson or Negative Binomial regression

Recency, frequency, monetary (RFM) features

- **RFM** - approach that underlies many feature engineering methods
- **Recency** - time since last customer transaction
- **Frequency** - number of purchases in the observed period
- **Monetary** value - total amount spent in the observed period

Explore the sales distribution by month

```
# Explore monthly distribution of observations
online.groupby(['InvoiceMonth']).size()
```

```
InvoiceMonth
2010-12    4893
2011-01    3580
2011-02    3648
2011-03    4764
2011-04    4148
2011-05    5018
2011-06    4669
2011-07    4610
2011-08    4744
2011-09    7189
2011-10    8808
2011-11    9513
dtype: int64
```

Separate feature data

```
# Exclude target variable
online_X = online[online['InvoiceMonth']!='2011-11']

# Define snapshot date
NOW = dt.datetime(2011,11,1)

# Build the features
features = online_X.groupby('CustomerID').agg({
                                'InvoiceDate': lambda x: (NOW - x.max()).days,
                                'InvoiceNo': pd.Series.nunique,
                                'TotalSum': np.sum,
                                'Quantity': ['mean', 'sum']
                                }).reset_index()

features.columns = ['CustomerID', 'recency', 'frequency',
                    'monetary', 'quantity_avg', 'quantity_total']
```

Review features

```
print(features.head())
```

	CustomerID	recency	frequency	monetary	quantity_avg	quantity_total
0	12747.0	27	9	643.33	10.523810	221
1	12748.0	5	107	4576.23	6.727110	3747
2	12749.0	91	2	598.65	8.791667	211
3	12820.0	5	3	202.62	10.307692	134
4	12822.0	31	2	146.15	9.666667	87

Calculate target variable

```
# Build pivot table with monthly transactions per customer
cust_month_tx = pd.pivot_table(data=online, index=['CustomerID'],
                               values='InvoiceNo',
                               columns=['InvoiceMonth'],
                               aggfunc=pd.Series.nunique, fill_value=0)
print(cust_month_tx.head())
```

InvoiceMonth	2010-12	2011-01	2011-02	2011-03	2011-04	2011-05	2011-06	2011-07	2011-08	2011-09	2011-10	2011-11
CustomerID												
12747.0	2	1	0	1	0	2	1	0	1	0	1	1
12748.0	24	2	4	9	3	17	12	8	9	9	10	41
12749.0	0	0	0	0	0	1	0	0	1	0	0	1
12820.0	0	1	0	0	0	0	0	0	0	1	1	0
12822.0	0	0	0	0	0	0	0	0	0	2	0	0

Finalize data preparation and split to train/test

```
# Store identifier and target variable column names
custid = ['CustomerID']
target = ['2011-11']

# Extract target variable
Y = cust_month_tx[target]

# Extract feature column names
cols = [col for col in features.columns if col not in custid]

# Store features
X = features[cols]
```

Split data to training and testing

```
# Randomly split 25% of the data to testing
from sklearn.model_selection import train_test_split
train_X, test_X, train_Y, test_Y = train_test_split(X, Y,
                                                    test_size=0.25,
                                                    random_state=99)

# Print shapes of the datasets
print(train_X.shape, train_Y.shape, test_X.shape, test_Y.shape)
```

```
(2529, 5) (2529, 1) (843, 5) (843, 1)
```

Let's work on data preparation exercises!

MACHINE LEARNING FOR MARKETING IN PYTHON

Predicting customer transactions

MACHINE LEARNING FOR MARKETING IN PYTHON

Karolis Urbonas
Head of Analytics & Science, Amazon

Modeling approach

- Linear regression to predict next month's transactions.
- Same modeling steps as with logistic regression.

Modeling steps

1. **Split** data to training and testing
2. **Initialize** the model
3. **Fit** the model on the training data
4. **Predict** values on the testing data
5. **Measure** model performance on testing data

Regression performance metrics

Key metrics:

- **Root mean squared error (RMSE)** - Square root of the average squared difference between prediction and actuals
- **Mean absolute error (MAE)** - Average absolute difference between prediction and actuals
- **Mean absolute percentage error (MAPE)** - Average percentage difference between prediction and actuals (actuals can't be zeros)

Additional regression and supervised learning metrics

- **R-squared** - statistical measure that represents the percentage proportion of variance that is explained by the model. Only applicable to regression, **not** classification. **Higher is better.**
- Coefficient **p-values** - probability that the regression (or classification) coefficient is observed due to chance. **Lower is better.** Typical thresholds are 5% and 10%.

Fitting the model

```
# Import the linear regression module
from sklearn.linear_model import LinearRegression

# Initialize the regression instance
linreg = LinearRegression()

# Fit model on the training data
linreg.fit(train_X, train_Y)

# Predict values on both training and testing data
train_pred_Y = linreg.predict(train_X)
test_pred_Y = linreg.predict(test_X)
```

Measuring model performance

```python
# Import performance measurement functions
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
# Calculate metrics for training data
rmse_train = np.sqrt(mean_squared_error(train_Y, train_pred_Y))
mae_train = mean_absolute_error(train_Y, train_pred_Y)
# Calculate metrics for testing data
rmse_test = np.sqrt(mean_squared_error(test_Y, test_pred_Y))
mae_test = mean_absolute_error(test_Y, test_pred_Y)
# Print performance metrics
print('RMSE train: {:.3f}; RMSE test: {:.3f}\nMAE train: {:.3f}, MAE test: {:.3f}'.format(
                            rmse_train, rmse_test, mae_train, mae_test))
```

```
RMSE train: 0.717; RMSE test: 1.216
MAE train: 0.514, MAE test: 0.555
```

Interpreting coefficients

- Need to assess statistical significance
- Introduction to `statsmodels` library
- Gives in-depth model summary

Build regression model with statsmodels

```python
# Import the library
import statsmodels.api as sm

# Convert target variable to `numpy` array
train_Y = np.array(train_Y)

# Initialize and fit the model
olsreg = sm.OLS(train_Y, train_X)
olsreg = olsreg.fit()

# Print model summary
print(olsreg.summary())
```

Regression summary table

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.488
Model:                            OLS   Adj. R-squared:                  0.487
Method:                 Least Squares   F-statistic:                     480.3
Date:                Sun, 18 Aug 2019   Prob (F-statistic):               0.00
Time:                        17:03:53   Log-Likelihood:                -2769.8
No. Observations:                2529   AIC:                             5550.
Df Residuals:                    2524   BIC:                             5579.
Df Model:                           5
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
recency            0.0002      0.000      1.701      0.089   -2.92e-05       0.000
frequency          0.1316      0.003     38.000      0.000       0.125       0.138
monetary        1.001e-06   3.59e-05      0.028      0.978   -6.95e-05    7.15e-05
quantity_avg       0.0001      0.000      0.803      0.422      -0.000       0.000
quantity_total    -0.0001   5.74e-05     -2.562      0.010      -0.000   -3.45e-05
==============================================================================
```

Interpreting R-squared

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.488
Model:                            OLS   Adj. R-squared:                  0.487
Method:                 Least Squares   F-statistic:                     480.3
Date:                Sun, 18 Aug 2019   Prob (F-statistic):               0.00
Time:                        17:03:53   Log-Likelihood:                -2769.8
No. Observations:                2529   AIC:                             5550.
Df Residuals:                    2524   BIC:                             5579.
Df Model:                           5
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
recency            0.0002      0.000      1.701      0.089   -2.92e-05       0.000
frequency          0.1316      0.003     38.000      0.000       0.125       0.138
monetary        1.001e-06   3.59e-05      0.028      0.978   -6.95e-05    7.15e-05
quantity_avg       0.0001      0.000      0.803      0.422      -0.000       0.000
quantity_total    -0.0001   5.74e-05     -2.562      0.010      -0.000   -3.45e-05
==============================================================================
```

Interpreting coefficient p-values

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.488
Model:                            OLS   Adj. R-squared:                  0.487
Method:                 Least Squares   F-statistic:                     480.3
Date:                Sun, 18 Aug 2019   Prob (F-statistic):               0.00
Time:                        17:03:53   Log-Likelihood:                -2769.8
No. Observations:                2529   AIC:                             5550.
Df Residuals:                    2524   BIC:                             5579.
Df Model:                           5
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
recency            0.0002      0.000      1.701      0.089   -2.92e-05       0.000
frequency          0.1316      0.003     38.000      0.000       0.125       0.138
monetary        1.001e-06   3.59e-05      0.028      0.978   -6.95e-05    7.15e-05
quantity_avg       0.0001      0.000      0.803      0.422      -0.000       0.000
quantity_total    -0.0001   5.74e-05     -2.562      0.010      -0.000   -3.45e-05
==============================================================================
```

Let's build some regression models!

MACHINE LEARNING FOR MARKETING IN PYTHON