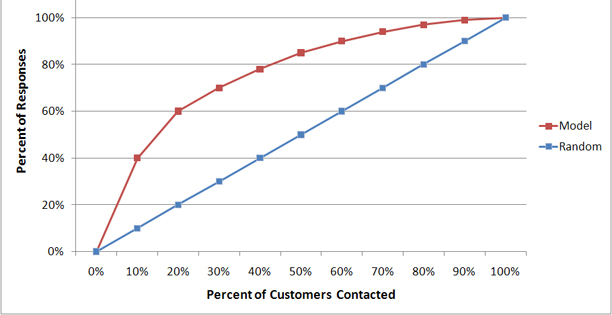
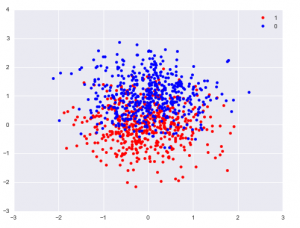
# Метрика качества: lift chart

[[](http://glebmikhaylov.com/%d0%bc%d0%b5%d1%82%d1%80%d0%b8%d0%ba%d0%b0-%d0%ba%d0%b0%d1%87%d0%b5%d1%81%d1%82%d0%b2%d0%b0-lift-chart/)](http://glebmikhaylov.com/%d0%bc%d0%b5%d1%82%d1%80%d0%b8%d0%ba%d0%b0-%d0%ba%d0%b0%d1%87%d0%b5%d1%81%d1%82%d0%b2%d0%b0-lift-chart/" \o "Метрика качества: lift chart" \t "_blank)

Lift chart - одна из метрик качества предсказательных моделей. Ее преимущество в том, что в отличии например от roc-кривой её очень легко презентовать и интерпретировать. Представьте, что у вас есть база клиентов, которую нужно обзвонить и что-то продать. Наверное выпостроите модель, которая будет предсказывать вероятность положительного отклика клиента и будете звонить клиентам в порядке убывания вероятности. Но как объяснить вашему заказчику, какое преимущество он получит, используя вашу модель? Lift chart позволит это очень хорошо объяснить. Давайте смоделируем какие-нибудь данные, постоим модель и проанализируем её. Я сгенерирую два класса, которые довольно трудно отделить один от другого. Всего у нас будет 1000 клиентов и только 500 из них откликнуться на наше предложение.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | importpandas aspd  importnumpy asnp  fromsklearn.linear\_model importLogisticRegression  fromsklearn.cross\_validation importtrain\_test\_split  fromsklearn.metrics importaccuracy\_score,roc\_auc\_score,roc\_curve  importmatplotlib.pyplot asplt  importseaborn  seaborn.set(font='arial')  %matplotlibinline  np.random.seed(0)  data\_0=np.random.multivariate\_normal([0,0],[[0.5,0],[0,0.5]],size=500)  data\_1=np.random.multivariate\_normal([0,1],[[0.5,0],[0,0.5]],size=500)  plt.figure(figsize=(8,6))  plt.scatter(data\_0[:,0],data\_0[:,1],color='red',label='1')  plt.scatter(data\_1[:,0],data\_1[:,1],color='blue',label='0')  plt.legend() |



Синий цвет - нет отклика, красный цвет - есть отклик. Создадим обучающую и тестовую выборки.

|  |  |
| --- | --- |
| 1  2  3 | X=np.vstack([data\_0,data\_1])  y=np.hstack([np.zeros(500),np.ones(500)])  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y) |

Обучим логистическую регрессию и посмотрим на её качество.

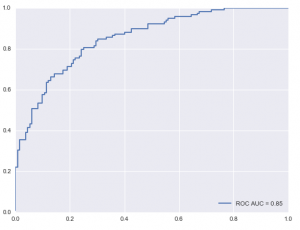
|  |  |
| --- | --- |
| 1  2  3  4  5 | regressor=LogisticRegression()  regressor.fit(X\_train,y\_train)  y\_pred=regressor.predict(X\_test)# порог 0.5  y\_pred\_proba=regressor.predict\_proba(X\_test)[:,1] |

Посмотрим на точность.

|  |  |
| --- | --- |
| 1  2 | printaccuracy\_score(y\_test,y\_pred)  # 0.756 |

Построим roc-кривую.

|  |  |
| --- | --- |
| 1  2  3  4  5 | roc\_auc=roc\_auc\_score(y\_test,y\_pred\_proba)  tpr,fpr,thr=roc\_curve(y\_test,y\_pred\_proba)  plt.figure(figsize=(8,6))  plt.plot(tpr,fpr,label='ROC AUC = {:.2f}'.format(roc\_auc))  plt.legend(loc='lower right') |



Получив эти значения ([про метрики качества можно тоже у меня почитать](http://glebmikhaylov.com/%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B8-%D0%BA%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%B0/)) вы бежите к заказчику и говорите: "Смотри, как круто! Надо срочно начинать обзвон!" Заказчик говорит: "Хм, классно. Я ничего не понимаю. На сколько это лучше, чем просто звонить всем подряд?" Вот тут-то вам и пригодиться lift chart.

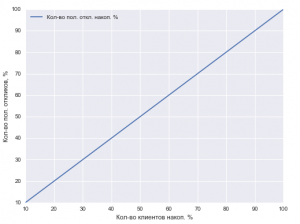
Давайте как-нибудь визуализируем случайный обзвон. Представим, что сам обзвон производиться в десять подходов. Какую долю положительных откликов мы будем иметь на каждом этапе? Проиллюстрирую это таблицей для нашего случая.

|  |  |
| --- | --- |
| 1  2  3 | df=pd.DataFrame({u'Кол-во клиентов':[100]\*10,  u'Кол-во положительных откликов':[100\*0.5]\*10})  df |



Теперь посчитаем накопительных итог по каждому столбцу в процентах от общей суммы и выведем график.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | df[u'Кол-во клиентов накоп. %']=100\*df[u'Кол-во клиентов'].cumsum()/\  df[u'Кол-во клиентов'].sum()  df[u'Кол-во пол. откл. накоп. %']=100\*df[u'Кол-во положительных откликов'].cumsum()/\  df[u'Кол-во положительных откликов'].sum()  df.plot(u'Кол-во клиентов накоп. %',u'Кол-во пол. откл. накоп. %',figsize=(8,6))  plt.ylabel(u'Кол-во пол. откликов, %') |



Интерпретировать нужно так: обзвонив 10% клиентов, мы получим 10% положительных откликов (из всех возможных), обзвонив 20% - 20% и т.д. Теперь пришло время построить подобный график для нашей модели. Для этого нужно отсортировать клиентов в порядке убывания вероятности отклика и сравнить в реальным откликом (по тестовой выборке).

|  |  |
| --- | --- |
| 1  2  3 | df=pd.DataFrame({u'Вероятность отклика':y\_pred\_proba,u'Реальный отклик':y\_test})  df.sort\_values(u'Вероятность отклика',ascending=False,inplace=True)  df.head() |



Далее нужно сделать тоже самое, что и в случае со случайным обзвоном. Сначала посчитаем кол-во клиентов, которым позвонили, накопленным итогом в процентах.

|  |  |
| --- | --- |
| 1  2  3 | df[u'Кол-во клиентов']=1  df[u'Кол-во клиентов накоп. %']=100\*df[u'Кол-во клиентов'].cumsum()/df[u'Кол-во клиентов'].sum()  df.head() |



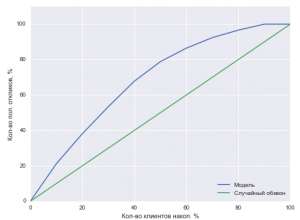
Побьем их на децили (т.е. на десять равных частей), сгруппируем и просуммируем (добавим еще строчку с [0,0], чтобы график смотрелся более красиво).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | decile=pd.cut(df[u'Кол-во клиентов накоп. %'],  np.arange(0,1.1,0.1)\*100,  labels=np.arange(0.1,1.1,0.1)\*100)  df=df.groupby(decile)[u'Реальный отклик'].sum().reset\_index()  df=pd.DataFrame(np.concatenate([[[0,0]],df.values,]),columns=df.columns)  df |



И теперь осталось только посчитать накопленный итог в процентах по реальному отклику. И построить график. На графике сразу же отобразим график случайной модели.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | df[u'Реальный отклик накоп. %']=100\*df[u'Реальный отклик'].cumsum()/df[u'Реальный отклик'].sum()  df[u'Кол-во клиентов накоп. %']=df[u'Кол-во клиентов накоп. %'].astype('float64')  df.plot(u'Кол-во клиентов накоп. %',  u'Реальный отклик накоп. %',  xlim=[0,100],ylim=[0,110],  label=u'Модель',figsize=(8,6))  plt.plot(df[u'Кол-во клиентов накоп. %'],  df[u'Кол-во клиентов накоп. %'],  label=u'Случайный обзвон')  plt.legend(loc='lower right')  plt.ylabel(u'Кол-во пол. откликов, %') |



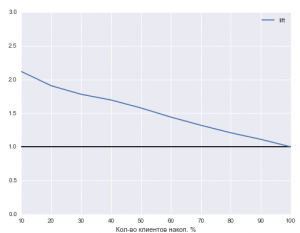
На этой диаграмме видно, какое преимущество дает наша модель на каждом этапе обзвона. Например можно увидеть, что обзвонив только 20% базы, мы получаем почти 40% всех положительных откликов, т.е. в два раза больше чем при случайном обзвоне. Теперь можно построить и сам lift chart. Для этого расчитаем lift, т.е отношение отклика по моделе и случайного обзвона.

|  |  |
| --- | --- |
| 1  2  3 | df=df.loc[1:]# нулевой дециль нам больше не нужен  df['lift']=df[u'Реальный отклик накоп. %']/(df[u'Кол-во клиентов накоп. %'])  df |



Построим график.

|  |  |
| --- | --- |
| 1  2 | df.plot(u'Кол-во клиентов накоп. %','lift',ylim=[0,3],figsize=(8,6))  plt.hlines(1,0,100) |



Теперь презентуя свою модель заказчику, вы можете объяснить ему, что при использовании вашей модели обзвовон будет в полтора раза эффективнее, чем без модели:

|  |  |
| --- | --- |
| 1  2 | df.lift.mean()  # 1.5155299973096583 |

Так же можно сказать, что обзвонив 50% базы, вы получите 80% всех положительных откликов:

|  |  |
| --- | --- |
| 1 | df[df[u'Кол-во клиентов накоп. %']==50] |

fifty_perc

Надеюсь, что статья понравилась. До новых встреч!)

## [Ссылка на ipython notebook.](https://github.com/glebmikha/ipython_notebooks/blob/master/Lift%20chart.ipynb)

[5](http://glebmikhaylov.com/%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D0%BA%D0%B0-%D0%BA%D0%B0%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%B0-lift-chart/)

Measure

Measure