

Лекція 3.

Моделювання відтоку клієнтів / Churn Prediction

Математичні моделі в продуктовому маркетингу

Що таке відтік клієнтів / Churn ?

Це коли існуючий клієнт, користувач, гравець, передплатник тощо припиняє відносини з компанією через:

- контрактний відтік - скасування підписки або призупинення контракту на отримання послуги, напр. послуг кабельного телебачення / передплатених послуг мобільного зв'язку, підписки Netflix / SaaS.
- неконтрактний відтік - коли клієнт не має контракту на послугу, включає лояльність споживачів у роздрібних магазинах або в e-commerce.
- мимовільний відтік - коли відтік відбувається не на вимогу клієнта, напр. через закінчення терміну дії кредитної картки, або через відключення комунальних послуг за несплату.

Причини контрактного відтоку клієнтів

- Недостатнє використання
- Погане обслуговування
- Краща ціна
- Нові технологічні рішення
- Вірусність продуктів-аналогів або агресивний маркетинг конкурентів
- Заміна продукту / послуги чимось іншим

Telco Churn Dataset : залежна змінна

Залежна змінна: Churn

Тут відтік визначається тим, що клієнт скасовує свій тарифний план стільникового зв'язку в певний момент часу, і кодується в наборі даних як «ні» або «так».

Description	Value
Records	3333
Features	21
Continuous	15
Categorical	6

Telco Churn Dataset : атрибути клієнтів

State: штат, в якому знаходиться клієнт.

Area_Code: телефонний код регіону, в якому знаходиться клієнт.

Phone: номер телефону клієнта.

Intenational_Plan: чи має клієнт міжнародний тарифний план (так/ні).

Voice_Mail_Plan: чи має клієнт тарифний план голосової пошти (так/ні).

Number_Vmail_Messages: кількість голосових повідомлень, які були залишені клієнтові за останній місяць.

Total_Day_Minutes: загальна тривалість дзвінків клієнта протягом дня (в хвилинах).

Total_Day_Calls: загальна кількість дзвінків, які клієнт здійснив протягом дня.

Total_Day_Charge: загальна вартість дзвінків клієнта протягом дня.

Total_Eve_Minutes: загальна тривалість дзвінків клієнта ввечері (в хвилинах).

Telco Churn Dataset : атрибути клієнтів

Total_Eve_Calls: загальна кількість дзвінків, які клієнт здійснив ввечері.

Total_Eve_Charge: загальна вартість дзвінків клієнта ввечері.

Total_Night_Minutes: загальна тривалість дзвінків клієнта вночі (в хвилинах).

Total_Night_Calls: загальна кількість дзвінків, які клієнт здійснив вночі.

Total_Night_Charge: загальна вартість дзвінків клієнта вночі.

Total_Intl_Minutes: загальна тривалість міжнародних дзвінків клієнта.

Total_Intl_Calls: загальна кількість міжнародних дзвінків, які клієнт здійснив.

Total_Intl_Charge: загальна вартість міжнародних дзвінків клієнта.

Account_Length: кількість днів, протягом яких клієнт користувався зв'язком.

CustServ_Calls: кількість дзвінків до служби підтримки.

Exploratory Data Analysis with pandas

1. Зрозумійте особливості вибірки даних та обчисліть зведену статистику

`df.info(), df.head(), df.describe(), df.mean()`

2. Визначте кількість клієнтів, що скасувала підписку

`print(telco['Churn'].value _ counts())`

3. Визначте аномалії та пропущені значення.

```
no      2850
yes      483
Name: Churn, dtype: int64
```

Exploratory Data Analysis with pandas

3. Визначте відмінності між тими клієнтами, які скасували підписку і залишилися:

- Чи дзвонять у службу підтримку клієнти, що скасували підписку частіше?
- Порівняйте різні штати за кількістю скасованих підписок
- Проведіть групування клієнтів за різними критеріями

Group by x and compute the standard deviation

```
df.groupby(['x']).std()
```

```
# Count the number of churners and non-churners by State
```

```
print(telco.groupby('State')['Churn'].value_counts())
```


Візуалізація даних з seaborn & matplotlib

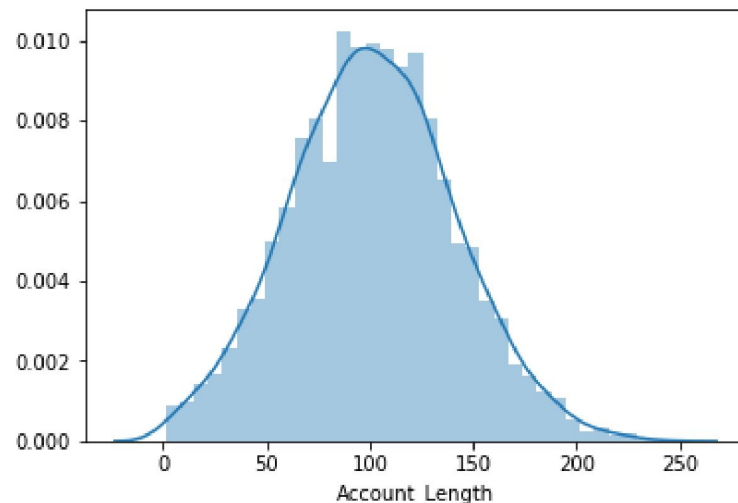
1. Візуалізація розподілу даних:

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
sns.distplot(telco['Account _ Length'])
```

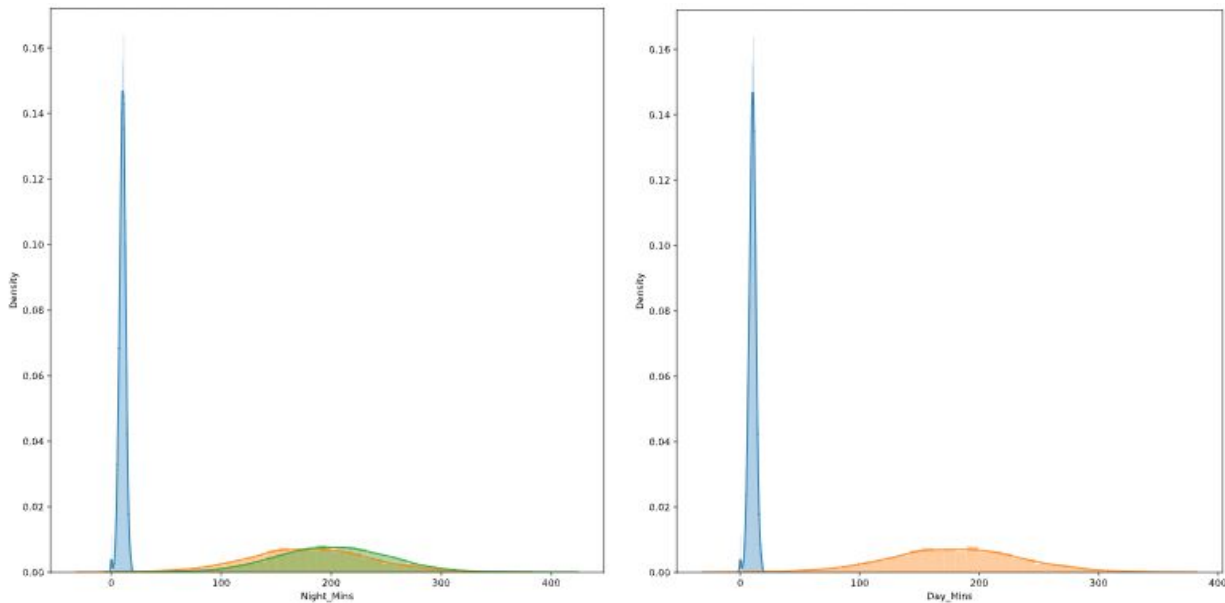
```
plt.show()
```



Візуалізація даних з seaborn & matplotlib

1. Visualize the distributions of other features using `seaborn`:

- `'Day_Mins'`
- `'Eve_Mins'`
- `'Night_Mins'`
- `'Intl_Mins'`

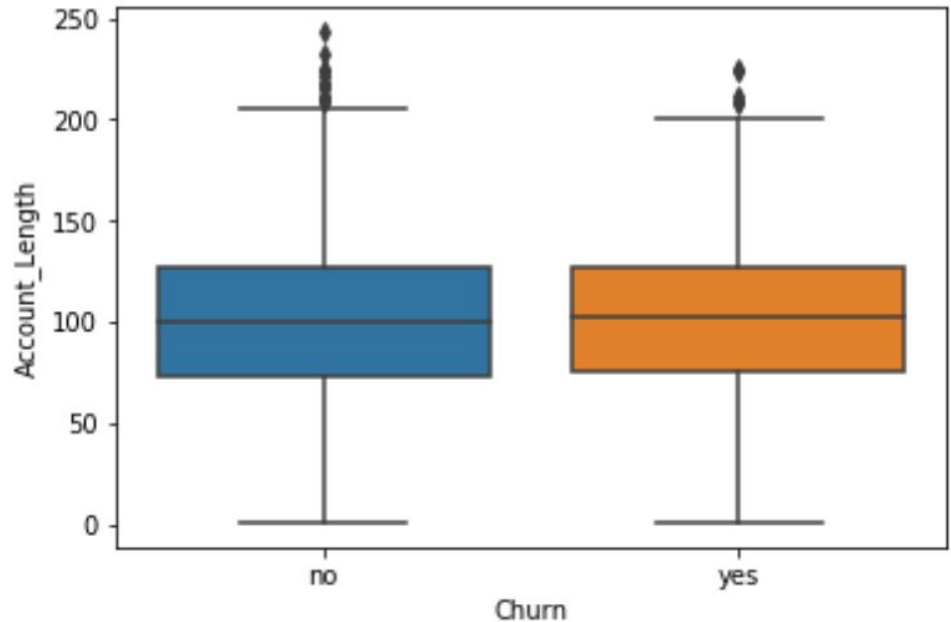


Візуалізація даних з seaborn & matplotlib

2. Відмінності в Account _ Length - Box plot:

```
sns.boxplot(x = 'Churn',  
            y = 'Account _ Length',  
            data = telco)  
  
plt.show()
```

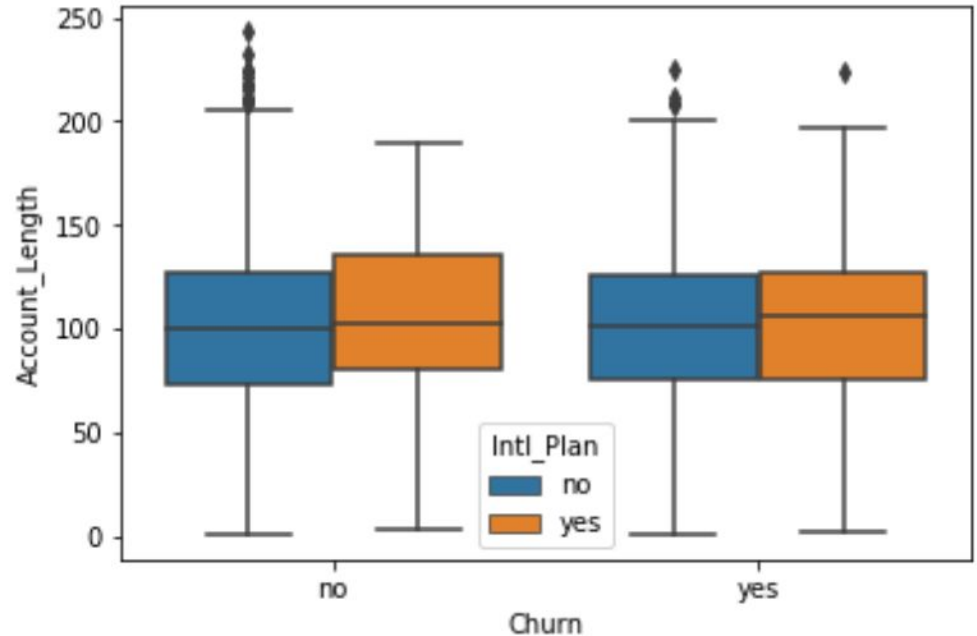
If you want to remove outliers,
you can specify the additional
parameter `sym=""`.



Візуалізація даних з seaborn & matplotlib

2. Відмінності в Account _ Length + Intl _ Plan - Box plot:

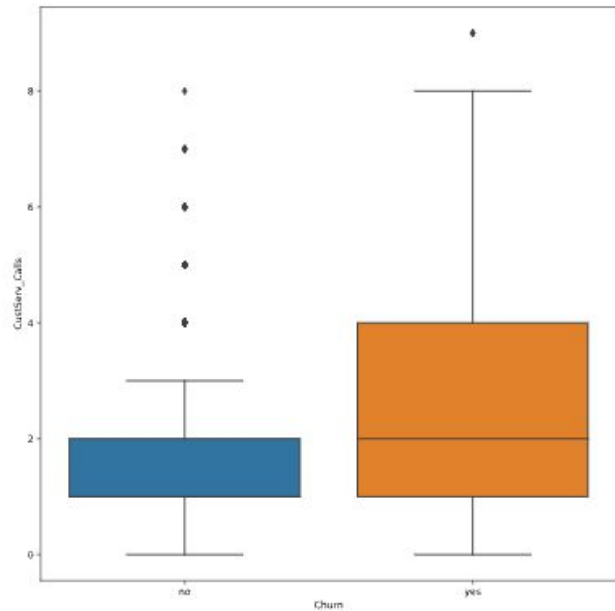
```
sns.boxplot(x = 'Churn',  
            y = 'Account _ Length',  
            data = telco,  
            hue = 'Intl _ Plan')  
plt.show()
```



Візуалізація даних з seaborn & matplotlib

2. Create a box plot with 'Churn' on the x-axis and 'CustServ_Calls' on the y-axis.

Дослідіть інші змінні.



Date Preparation / Підготовка даних

Model assumptions / Припущення моделей:

- змінні розподілені нормально
- змінні масштабовані в одній розмірній сітці

Типи даних:

- Алгоритми машинного навчання потребують числових типів даних
- Потрібно закодувати категоріальні змінні як числові

telco.dtype

Account_Length	int64
Vmail_Message	int64
Day_Mins	float64
Eve_Mins	float64
Night_Mins	float64
Intl_Mins	float64
CustServ_Calls	int64
Churn	object
Intl_Plan	object
Vmail_Plan	object

Day_Calls	int64
Day_Charge	float64
Eve_Calls	int64
Eve_Charge	float64
Night_Calls	int64
Night_Charge	float64
Intl_Calls	int64
Intl_Charge	float64
State	object
Area_Code	int64
Phone	object

Encoding binary features

```
telco['Intl _ Plan'].head()
```

```
0      no
1      no
2      no
3     yes
4     yes
Name: Intl_Plan, dtype: object
```


Encoding binary features

Option 1: .replace()

```
telco['Intl _ Plan'].replace({'no':0 , 'yes':1})
```

```
telco['Intl _ Plan'].head()
```

Option 2: LabelEncoder()

```
from sklearn.preprocessing import LabelEncoder
```

```
LabelEncoder().fit _ transform(telco["Intl _ Plan"])
```

```
telco['Intl _ Plan'].head()
```

```
0    0
1    0
2    0
3    1
4    1
Name: Intl_Plan
```

Encoding state

```
telco['State'].head(4)
```

- Could assign a number to each state
- Bad idea
- Would make your model less effective

```
0    KS
1    OH
2    NJ
3    OH
Name: State, dtype: object
```

```
0    0
1    1
2    2
3    1
Name: State, dtype: int64
```

One hot encoding

State		State_KS	State_OH	State_NJ
KS		1	0	0
OH	➡	0	1	0
NJ		0	0	1
OH		0	1	0

Feature scaling

- Змінні мають бути однієї розмірності / масштабу
- Рідко трапляється на практиці

```
telco['Intl _ Calls'].describe()
```

```
count    3333.000000
mean      4.479448
std       2.461214
min       0.000000
25%       3.000000
50%       4.000000
75%       6.000000
max       20.000000
Name: Intl_Calls, dtype: float64
```

```
telco['Night _ Mins'].describe()
```

```
count    3333.000000
mean     200.872037
std      50.573847
min      23.200000
25%     167.000000
50%     201.200000
75%     235.300000
max     395.000000
Name: Night_Mins, dtype: float64
```

Standardization

- Центрує розподіл навколо середнього
- Обраховує кількість STD (ст. відхилень) від AVG для кожного спостереження

```
from sklearn.preprocessing import StandardScaler
```

```
df = StandardScaler().fit _ transform(df)
```

Dropping unnecessary features

- Унікальні ідентифікатори:
 - номери телефонів
 - Social security numbers
 - Account numbers

.drop() method

```
telco.drop(['Soc _ Sec' , 'Tax _ ID'], axis=1)
```

Drop 'Area_Code' and 'Phone' from telco as well.

Dropping correlated features

- Сильно корельовані змінні можна викинути
- Вони не надають додаткової інформації моделі

Dropping correlated features: *telco.corr()*

	Day_Mins	Eve_Mins	Night_Mins	Intl_Mins	CustServ_Calls	Day_Calls	Day_Charge	Eve_Calls	Eve_Charge	Night_Calls	Night_Charge	Intl_Calls	Intl_Charge
Day_Mins	1.000000	0.007043	0.004323	-0.010155	-0.013423	0.006750	1.000000	0.015769	0.007029	0.022972	0.004300	0.008033	-0.010092
Eve_Mins	0.007043	1.000000	-0.012584	-0.011035	-0.012985	-0.021451	0.007050	-0.011430	1.000000	0.007586	-0.012593	0.002541	-0.011067
Night_Mins	0.004323	-0.012584	1.000000	-0.015207	-0.009288	0.022938	0.004324	-0.002093	-0.012592	0.011204	0.999999	-0.012353	-0.015180
Intl_Mins	-0.010155	-0.011035	-0.015207	1.000000	-0.009640	0.021565	-0.010157	0.008703	-0.011043	-0.013605	-0.015214	0.032304	0.999993
CustServ_Calls	-0.013423	-0.012985	-0.009288	-0.009640	1.000000	-0.018942	-0.013427	0.002423	-0.012987	-0.012802	-0.009277	-0.017561	-0.009675
Day_Calls	0.006750	-0.021451	0.022938	0.021565	-0.018942	1.000000	0.006753	0.006462	-0.021449	-0.019557	0.022927	0.004574	0.021666
Day_Charge	1.000000	0.007050	0.004324	-0.010157	-0.013427	0.006753	1.000000	0.015769	0.007036	0.022972	0.004301	0.008032	-0.010094
Eve_Calls	0.015769	-0.011430	-0.002093	0.008703	0.002423	0.006462	0.015769	1.000000	-0.011423	0.007710	-0.002056	0.017434	0.008674
Eve_Charge	0.007029	1.000000	-0.012592	-0.011043	-0.012987	-0.021449	0.007036	-0.011423	1.000000	0.007596	-0.012601	0.002541	-0.011074
Night_Calls	0.022972	0.007586	0.011204	-0.013605	-0.012802	-0.019557	0.022972	0.007710	0.007596	1.000000	0.011188	0.000305	-0.013630
Night_Charge	0.004300	-0.012593	0.999999	-0.015214	-0.009277	0.022927	0.004301	-0.002056	-0.012601	0.011188	1.000000	-0.012329	-0.015186
Intl_Calls	0.008033	0.002541	-0.012353	0.032304	-0.017561	0.004574	0.008032	0.017434	0.002541	0.000305	-0.012329	1.000000	0.032372
Intl_Charge	-0.010092	-0.011067	-0.015180	0.999993	-0.009675	0.021666	-0.010094	0.008674	-0.011074	-0.013630	-0.015186	0.032372	1.000000

Dropping correlated features: *telco.corr()*

	Day_Mins	Eve_Mins	Night_Mins	Intl_Mins	CustServ_Calls	Day_Calls	Day_Charge	Eve_Calls	Eve_Charge	Night_Calls	Night_Charge	Intl_Calls	Intl_Charge
Day_Mins	1.000000	0.007043	0.004323	-0.010155	-0.013423	0.006750	1.000000	0.015769	0.007029	0.022972	0.004300	0.008033	-0.010092
Eve_Mins	0.007043	1.000000	-0.012584	-0.011035	-0.012985	-0.021451	0.007050	-0.011430	1.000000	0.007586	-0.012593	0.002541	-0.011067
Night_Mins	0.004323	-0.012584	1.000000	-0.015207	-0.009288	0.022938	0.004324	-0.002093	-0.012592	0.011204	0.999999	-0.012353	-0.015180
Intl_Mins	-0.010155	-0.011035	-0.015207	1.000000	-0.009640	0.021565	-0.010157	0.008703	-0.011043	-0.013605	-0.015214	0.032304	0.999993
CustServ_Calls	-0.013423	-0.012985	-0.009288	-0.009640	1.000000	-0.018942	-0.013427	0.002423	-0.012987	-0.012802	-0.009277	-0.017561	-0.009675
Day_Calls	0.006750	-0.021451	0.022938	0.021565	-0.018942	1.000000	0.006753	0.006462	-0.021449	-0.019557	0.022927	0.004574	0.021666
Day_Charge	1.000000	0.007050	0.004324	-0.010157	-0.013427	0.006753	1.000000	0.015769	0.007036	0.022972	0.004301	0.008032	-0.010094
Eve_Calls	0.015769	-0.011430	-0.002093	0.008703	0.002423	0.006462	0.015769	1.000000	-0.011423	0.007710	-0.002056	0.017434	0.008674
Eve_Charge	0.007029	1.000000	-0.012592	-0.011043	-0.012987	-0.021449	0.007036	-0.011423	1.000000	0.007596	-0.012601	0.002541	-0.011074
Night_Calls	0.022972	0.007586	0.011204	-0.013605	-0.012802	-0.019557	0.022972	0.007710	0.007596	1.000000	0.011188	0.000305	-0.013630
Night_Charge	0.004300	-0.012593	0.999999	-0.015214	-0.009277	0.022927	0.004301	-0.002056	-0.012601	0.011188	1.000000	-0.012329	-0.015186
Intl_Calls	0.008033	0.002541	-0.012353	0.032304	-0.017561	0.004574	0.008032	0.017434	0.002541	0.000305	-0.012329	1.000000	0.032372
Intl_Charge	-0.010092	-0.011067	-0.015180	0.999993	-0.009675	0.021666	-0.010094	0.008674	-0.011074	-0.013630	-0.015186	0.032372	1.000000

Feature engineering

- Створення нових функцій для покращення продуктивності моделі
- Доменна експертиза

Приклади інжинірингу змінних:

- Total Minutes: Sum of `Day_ Mins` , `Eve _ Mins` , `Night _ Mins` , `Intl _ Mins`
- Ratio between Minutes and Charge:

`telco['Day_ Cost'] = telco['Day_ Mins'] / telco['Day_ Charge']`

- Create a new feature - `'Avg_Night_Calls' = 'Night_Mins' / 'Night_Calls'`

Making Predictions

(Supervised) Machine Learning Brief :

- Goal: Predict whether or not a customer will churn
- Target Variable: 'Churn'
- Supervised Machine Learning
- Learn from historical (training) data to make new predictions

Model Selection

1. Logistic regression: Good baseline:
 - Переваги: простота та інтерпретація
 - Недоліки: погано пояснює складні зв'язки
2. Random forests
3. Support vector machines

Training your Model: Support Vector Classification

```
from sklearn.svm import SVC
```

```
svc = SVC()
```

```
svc.fit(telco[features], telco['target'])
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

```
prediction = svc.predict(new_customer)    print(prediction)
```

Accuracy

One possible metric: Accuracy

Total Number of Correct Predictions / Total Number of Data Points

What data to use?

Training data not representative of new data


Training and Test Sets using scikit-learn

- Fit your classifier to the training set
- Make predictions using the test set

```
from sklearn.model _ selection import train _ test _ split  
X _ train, X _ test, y _ train, y _ test = train _ test _ split(telco['data'],  
telco['target'], test _ size=0.2, random _ state = 42)  
from sklearn.svm import SVC  
svc = SVC()  
svc.fit(X _ train, y _ train)  
svc.predict(X _ test)
```

Computing Accuracy

```
svc.score(X_test, y_test)
```



0.857

85.7% accuracy: хороший старт для початку

Improving your model

Overfitting: Model fits the training data too closely

Underfitting: Does not capture trends in the training data

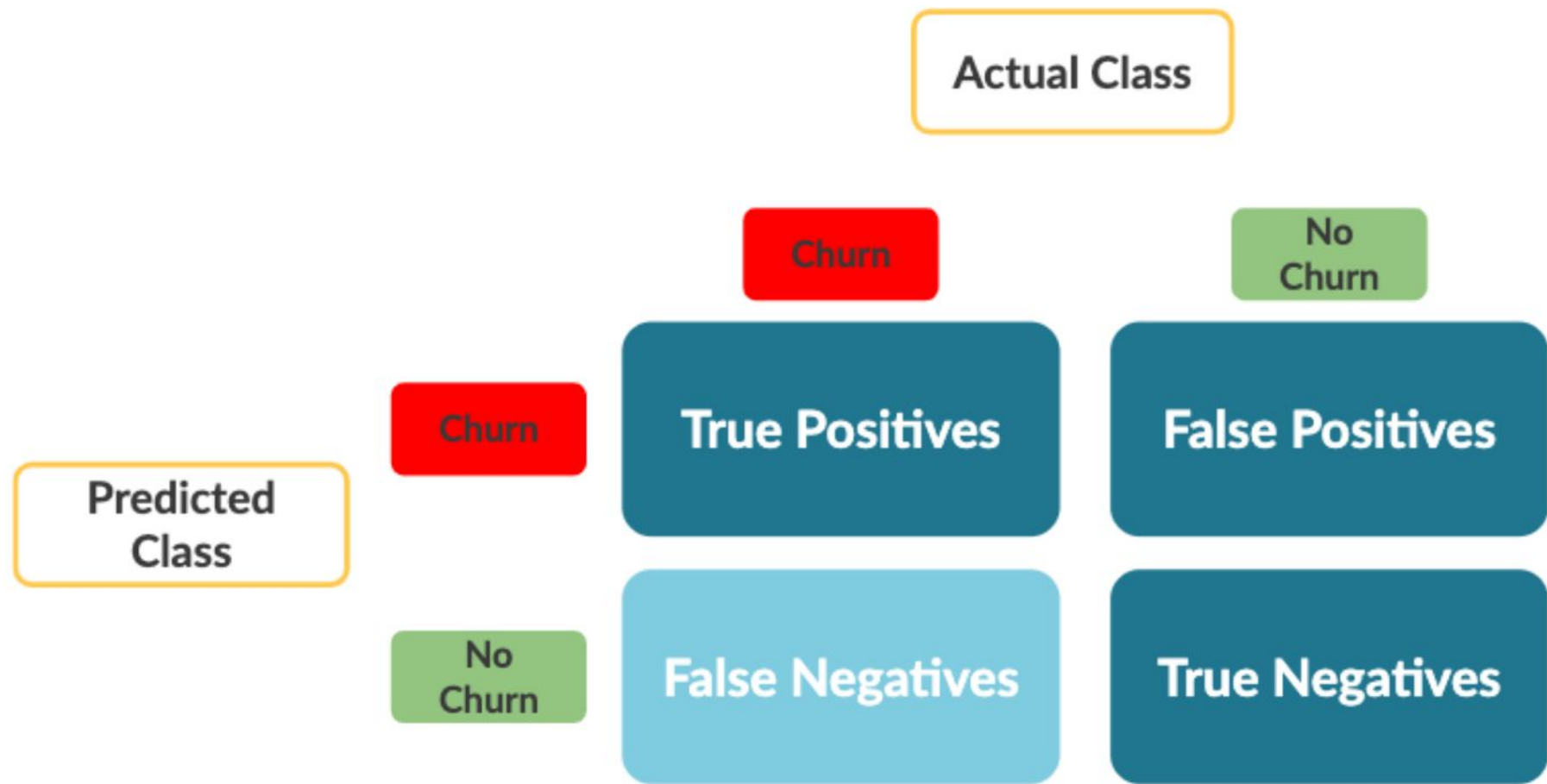
Need to find the right balance between Overfitting & Underfitting

Imbalanced classes

```
telco['Churn'].value_counts()
```

```
no      2850  
yes      483  
Name: Churn, dtype: int64
```

Accuracy not a very useful metric



Precision

Metric	Formula
Precision	$\text{True Positives} / (\text{True Positives} + \text{False Positives})$

A model with **high precision** indicates:

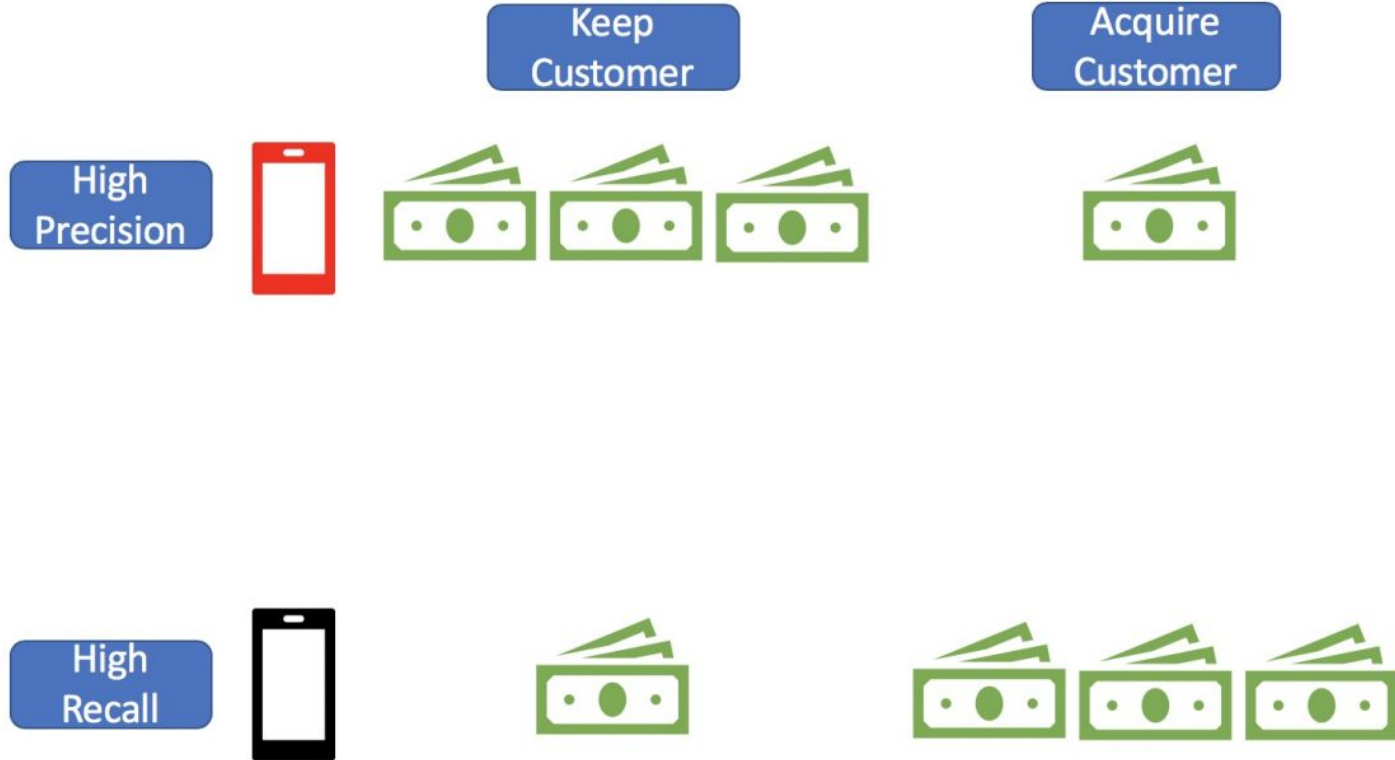
- Few false positives ("false alarms")
- Not many non-churners were classified as churners

Recall

Metric	Formula
Recall/Sensitivity	$\text{True Positives} / (\text{True Positives} + \text{False Negatives})$

A model with high recall indicates that it correctly classified most churners

Precision vs. Recall



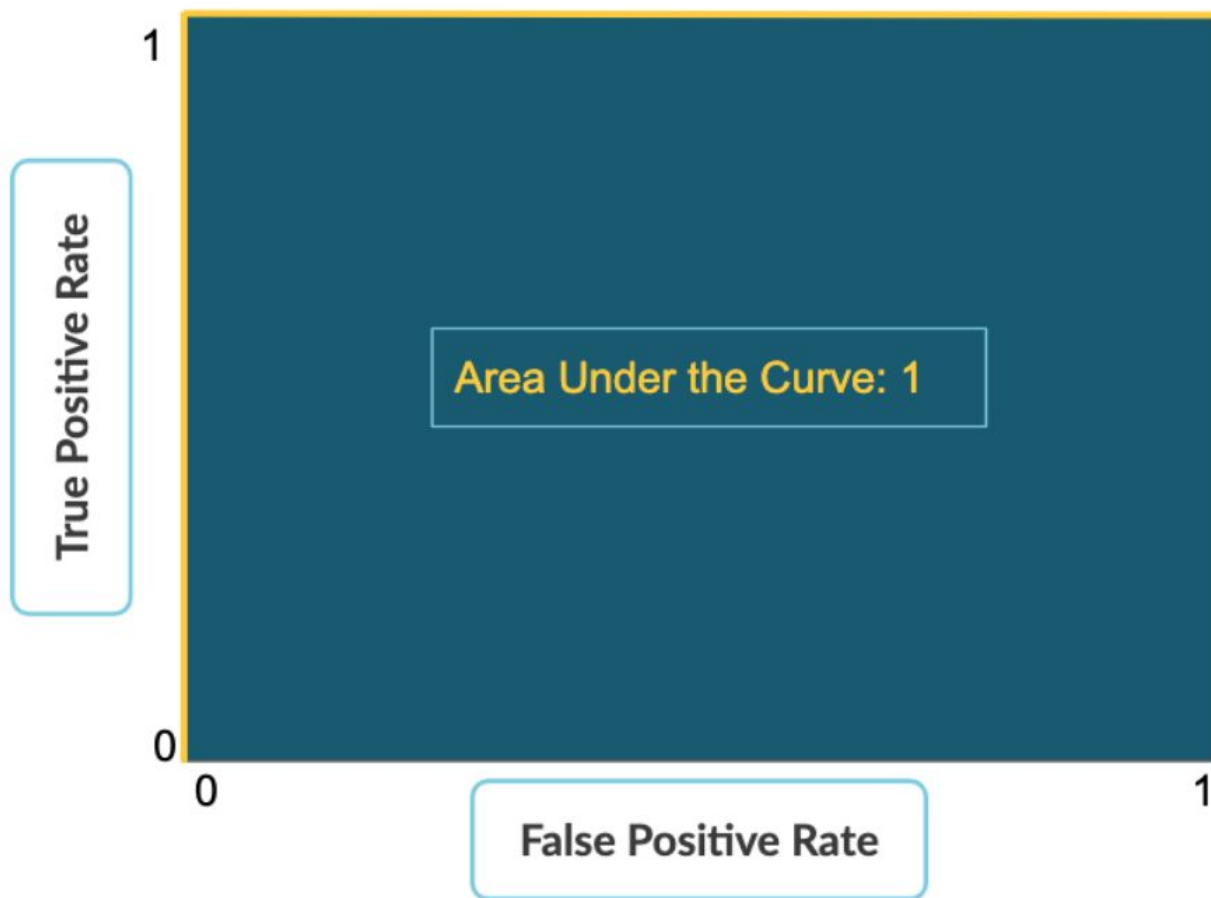
Confusion Matrix in scikit-learn

```
from sklearn.metrics import confusion _ matrix  
cm = confusion _ matrix(y_ test, y_ pred)
```

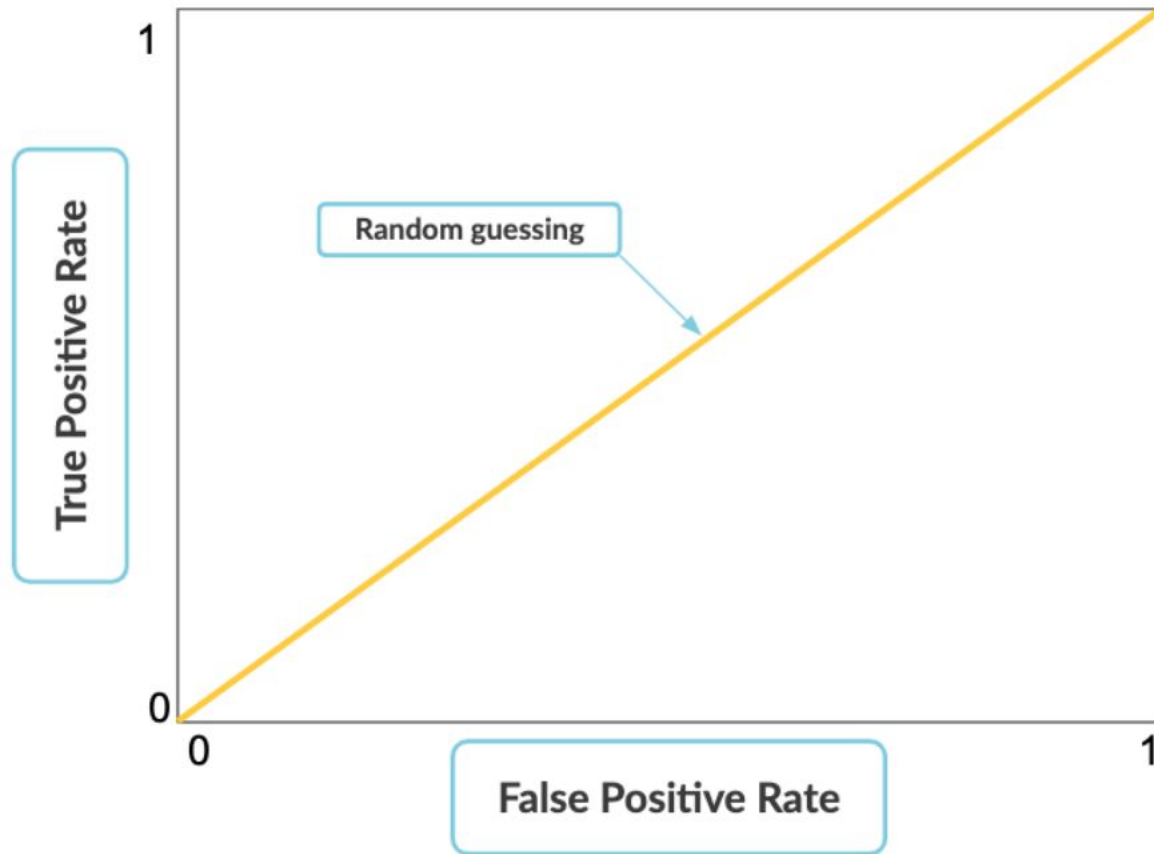
Other model metrics: Probability thresholds

- Every prediction your classifier makes has an associated probability
- Default probability threshold in scikit-learn: 50%
- What if we vary this threshold?

ROC Curve

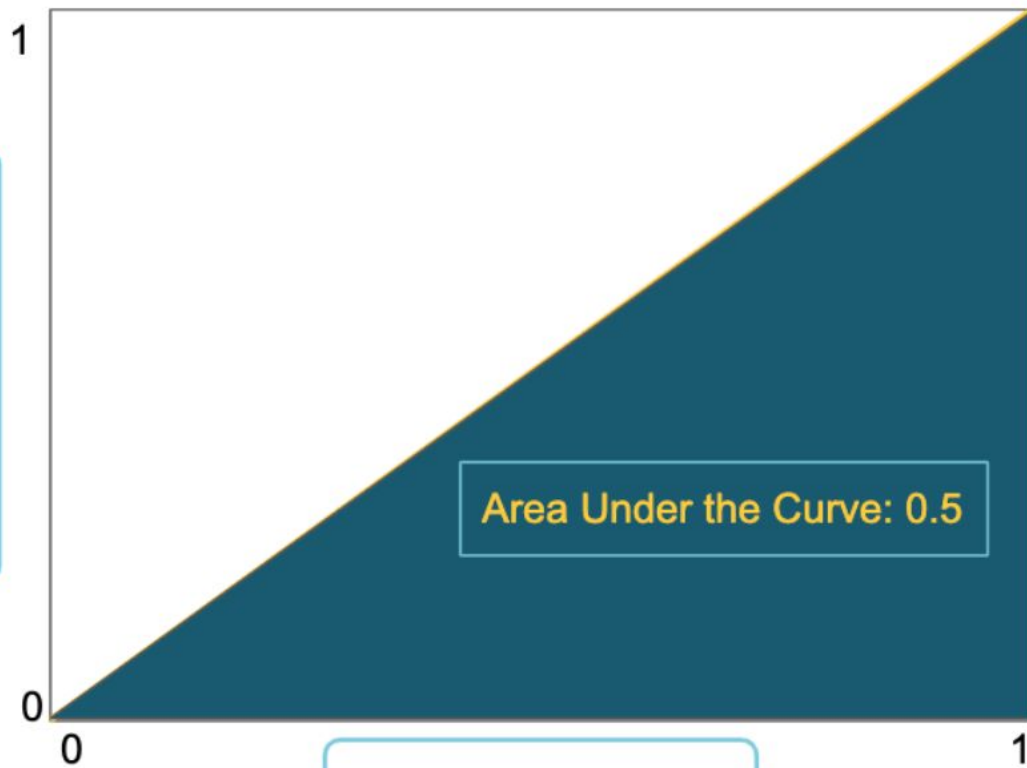


ROC Curve



ROC Curve

True Positive Rate



Area Under the Curve: 0.5

False Positive Rate

Generating probabilities in sklearn

```
logreg.predict_proba(X_test)[: , 1]
```

```
array([[0.80188981, 0.19811019],  
       [0.96484075, 0.03515925],  
       [0.9182671 , 0.0817329 ],  
       ...,
```

```
y_pred_prob = logreg.predict_proba(X_test)[: , 1]
```

ROC curve in sklearn

```
from sklearn.metrics import roc_curve  
  
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)  
  
import matplotlib.pyplot as plt  
  
plt.plot(fpr, tpr)  
  
plt.xlabel("False Positive Rate")  
  
plt.ylabel("True Positive Rate")  
  
plt.plot([0, 1], [0, 1], "k-- ")      plt.show()
```

Area under the curve

```
from sklearn.metrics import roc_auc_score
```

```
auc = roc_auc_score(y_test, y_pred)
```

F1 score

$$\text{F1 Score} = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

Перевага оцінки F1 полягає в тому, що вона об'єднує precision and recall в один показник. Висока оцінка F1 є ознакою гарної ефективності моделі навіть у ситуаціях, коли у вас можуть бути незбалансовані класи.

У scikit-learn ви можете обчислити оцінку f-1 за допомогою функції `f1_score`.

Приклад розрахунку

```
# Instantiate the classifier
```

```
clf = RandomForestClassifier()
```

```
# Fit to the training data
```

```
clf.fit(X_train, y_train)
```

```
# Predict the labels of the test set
```

```
y_pred = clf.predict(X_test)
```

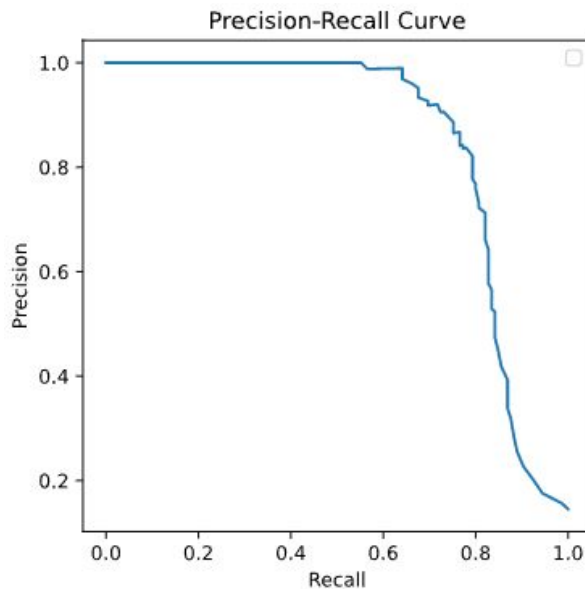
```
# Import f1_score
```

```
from sklearn.metrics import f1_score
```

```
# Print the F1 score
```

```
print(f1_score(y_test, y_pred))
```

```
<script.py> output: 0.7789473684210525
```



Tuning your model

Refresher:

```
from sklearn.svm import SVC
```

```
svc = SVC()
```

```
svc.fit(telco['data'], telco['target'])
```

Tuning your model: Random forest hyperparameters

Parameter	Purpose
n_estimators	Number of trees
criterion	Quality of Split
max_features	Number of features for best split
max_depth	Max depth of tree
min_sample_splits	Minimum samples to split node
bootstrap	Whether Bootstrap samples are used

Grid search in sklearn

```
from sklearn.model_selection import GridSearchCV

param_grid = {'n_estimators': np.arange(10, 51)}

clf_cv = GridSearchCV(RandomForestClassifier(), param_grid)

clf_cv.fit(X, y)

print(clf_cv.best_params_)

print(clf_cv.best_score_)

print(0.9237923792379238)
```

Feature importances

- Оцінює наскільки кожен атрибут (змінна) впливає на прогноз
- Ефективний спосіб донести результати до зацікавлених сторін
- Визначає які атрибути є важливими причинами відтоку?
- Визначає які атрибути можна видалити з моделі?

Interpretability vs accuracy

- Різні моделі мають різні прогностні якості
- Потрібно збалансувати точність прогнозу та можливість інтерпретації

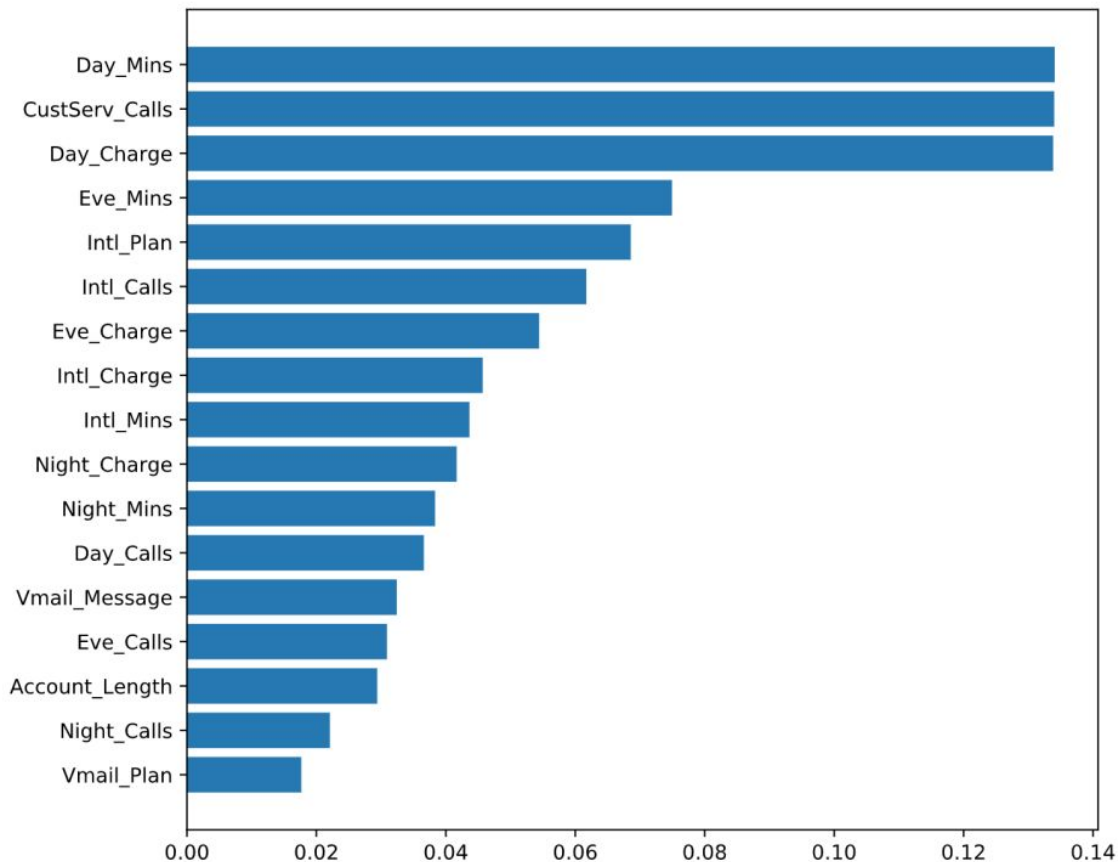
Random forest feature importances

```
random_forest = RandomForestClassifier()
```

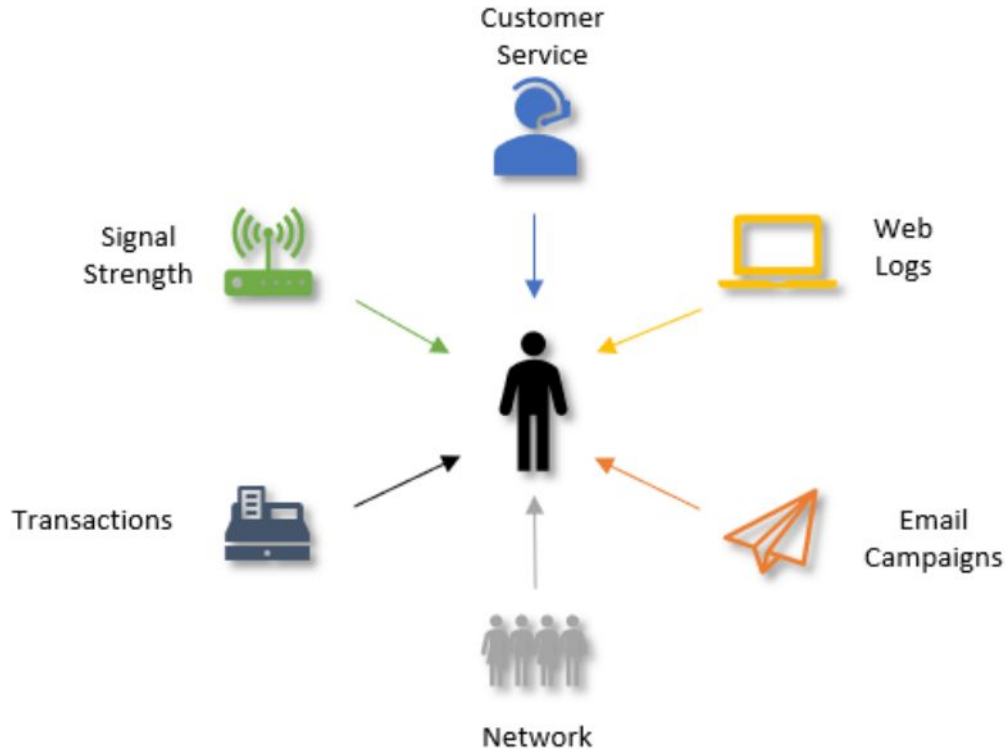
```
random_forest.fit(X_train, y_train)
```

```
random_forest.feature_importances_
```

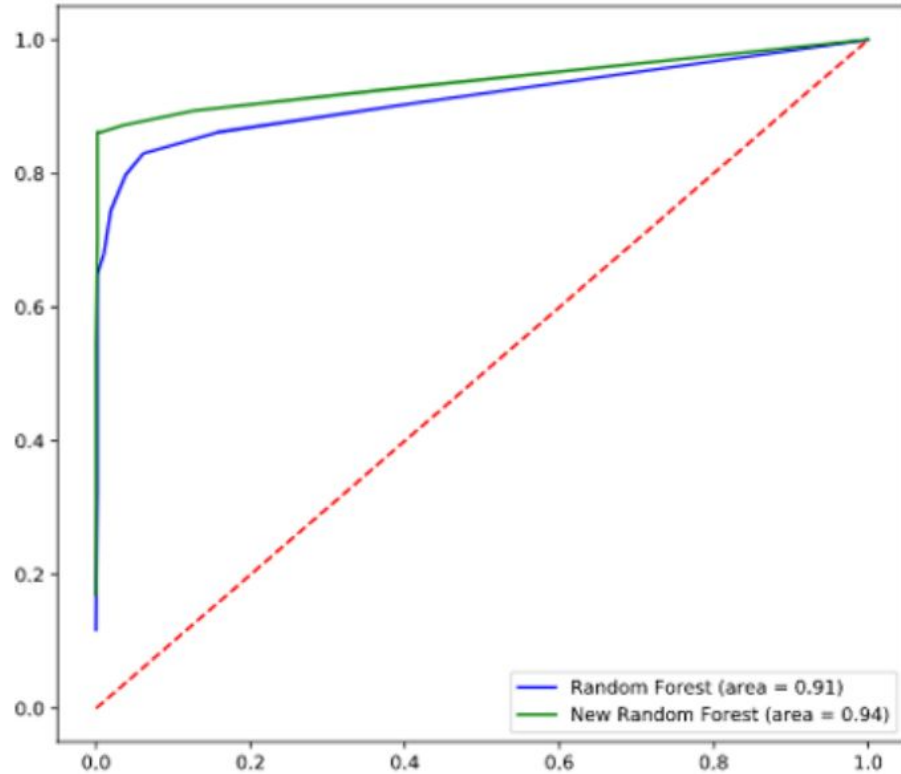
Random forest feature importances



Adding new data sources & features



Model Improvement



Next

Огляд моделей для прогнозування відтоку клієнтів (Churn Prediction):

- Logistic regression
- Decision Tree
- Random Forest