Project: Telco customer churn

Analyzing Preywart Chandra

Predicting **customer churn** is critical for **telecommunication companies** to be able to effectively retain customers. It is more costly to acquire new customers than to retain existing ones. For this reason, large telecommunications corporations are seeking to develop models to predict which customers are more likely to change and take actions accordingly.

In this article, we build a model to **predict how likely a customer will churn** by analyzing its characteristics:
(1) **demographic information**, (2) **account information**, and (3) **services information**. The objective is to obtain a data-driven solution that will allow us to reduce churn rates and, as a consequence, to increase customer satisfaction and corporation revenue.

Data set

The data set used in this article is available in the **Kaggle (**CC BY-NC-ND**)** and contains **nineteen columns (independent variables)** that indicate the **characteristics of the clients** of a fictional telecommunications corporation. The `Churn` column (**response variable**) indicates whether the customer departed within the last month or not. The class `No` includes the clients that did not leave the company last month, while the class `Yes` contains the clients that decided to terminate their relations with the company. The objective of the analysis is to obtain **the relation between the customer's characteristics and the churn**.

Steps of the project

The project consists of the following sections:

Data Reading

Exploratory Data Analysis and Data Cleaning

Data Visualization

Feature Importance

Feature Engineering

Setting a baseline

Splitting the data in training and testing sets

Assessing multiple algorithms

Algorithm selected: Gradient Boosting

Hyperparameter tuning

Performance of the model

Drawing conclusions — Summary

Show example of our file.

Reading and understanding the data

```
[ ] # Reading the dataset
    df = pd.read_csv('telecom_churn_data.csv')
    df.head()
```

	mobile_number	circle_id	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	last_date_of_month_6	last_date_of_month_7	last_date_of_month_8	last_date_of_month_9	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7	onnet_mou_
0	7000842753	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	197.385	214.816	213.803	21.100	NaN	NaN	0.0
1	7001865778	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	34.047	355.074	268.321	86.285	24.11	78.68	7.6
2	7001625959	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	167.690	189.058	210.226	290.714	11.54	55.24	37.2
3	7001204172	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	221.338	251.102	508.054	389.500	99.91	54.39	310.9
4	7000142493	109	0.0	0.0	0.0	6/30/2014	7/31/2014	8/31/2014	9/30/2014	261.636	309.876	238.174	163.426	50.31	149.44	83.8

Tag churners

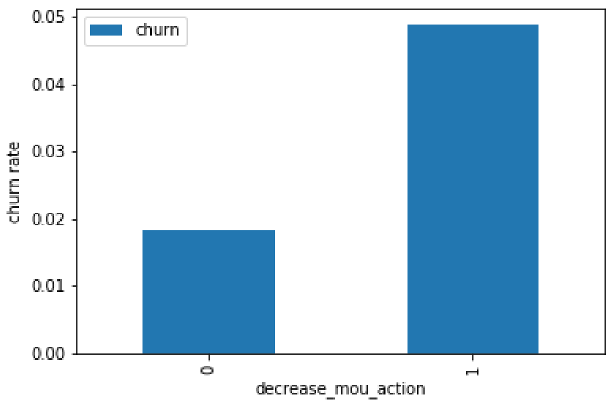
Now tag the churned customers (churn=1, else 0) based on the fourth month as follows: Those who have not made any calls (either incoming or outgoing) AND have not used mobile internet even once in the churn phase.

```
[ ] df['churn'] = np.where((df['total_ic_mou_9']==0) & (df['total_og_mou_9']==0) & (df['vol_2g_mb_9']==0) & (df['vol_3g_mb_9']==0), 1, 0)
```

```
[ ] df.head()
```

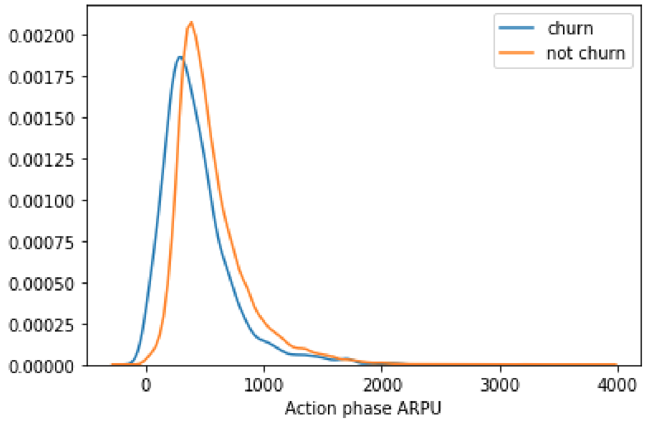
	mobile_number	loc_og_t2o_mou	std_og_t2o_mou	loc_ic_t2o_mou	arpu_6	arpu_7	arpu_8	arpu_9	onnet_mou_6	onnet_mou_7	onnet_mou_8
8	7001524846	0.0	0.0	0.0	378.721	492.223	137.362	166.787	413.69	351.03	35.08
13	7002191713	0.0	0.0	0.0	492.846	205.671	593.260	322.732	501.76	108.39	534.24
16	7000875565	0.0	0.0	0.0	430.975	299.869	187.894	206.490	50.51	74.01	70.61
17	7000187447	0.0	0.0	0.0	690.008	18.980	25.499	257.583	1185.91	9.28	7.79
21	7002124215	0.0	0.0	0.0	514.453	597.753	637.760	578.596	102.41	132.11	85.14

```
[ ] data.pivot_table(values='churn', index='decrease_mou_action', aggfunc='mean').plot.bar()
    plt.ylabel('churn rate')
    plt.show()
```

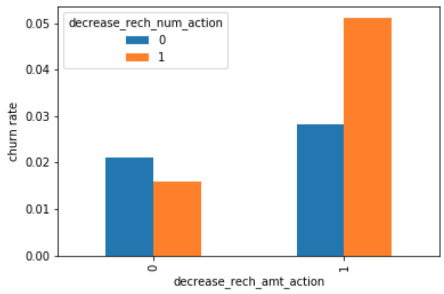


```python
# Distribution plot
ax = sns.distplot(data_churn['avg_arpu_action'],label='churn',hist=False)
ax = sns.distplot(data_non_churn['avg_arpu_action'],label='not churn',hist=False)
ax.set(xlabel='Action phase ARPU')
```

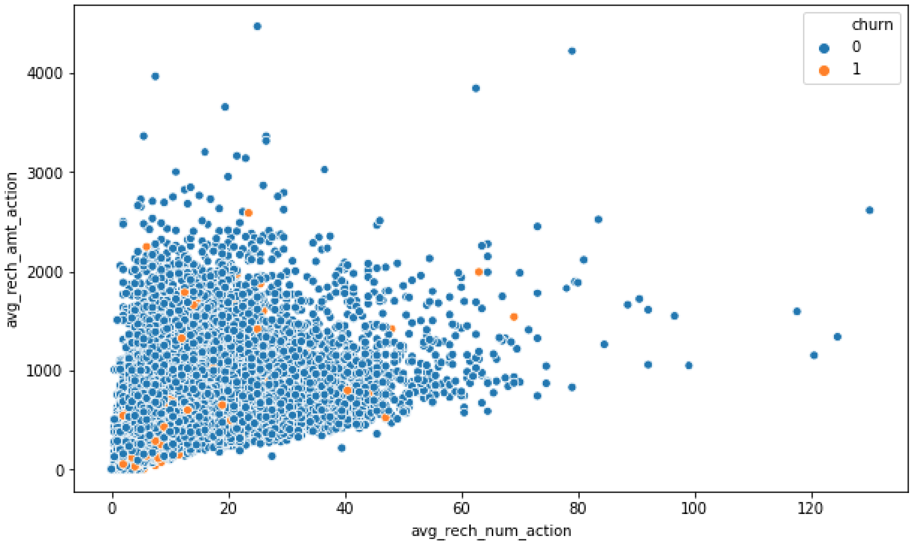
```
[Text(0.5, 0, 'Action phase ARPU')]
```




```python
data.pivot_table(values='churn', index='decrease_rech_amt_action', columns='decrease_rech_num_action', aggfunc='mean').plot.bar()
plt.ylabel('churn rate')
plt.show()
```

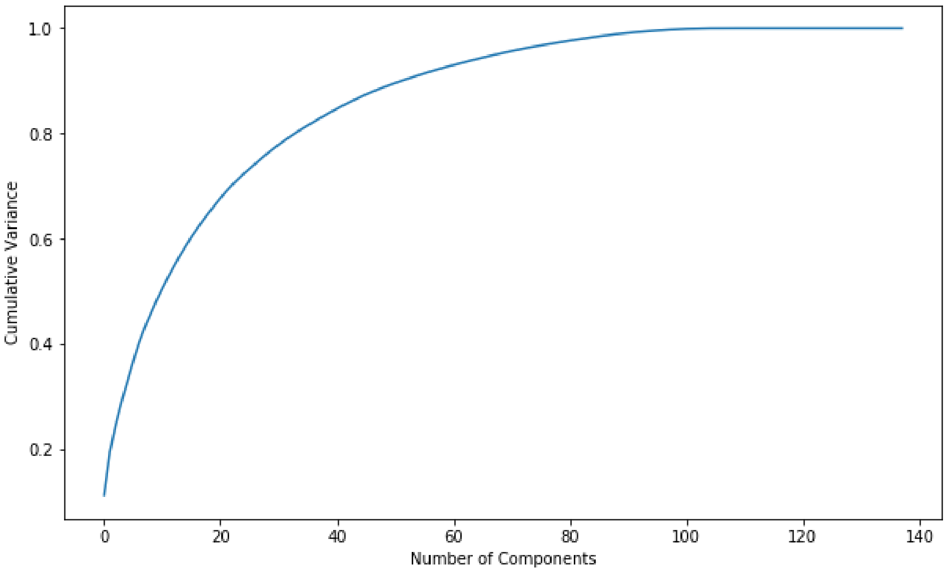


```
plt.figure(figsize=(10,6))
ax = sns.scatterplot('avg_rech_num_action','avg_rech_amt_action', hue='churn', data=data)
```

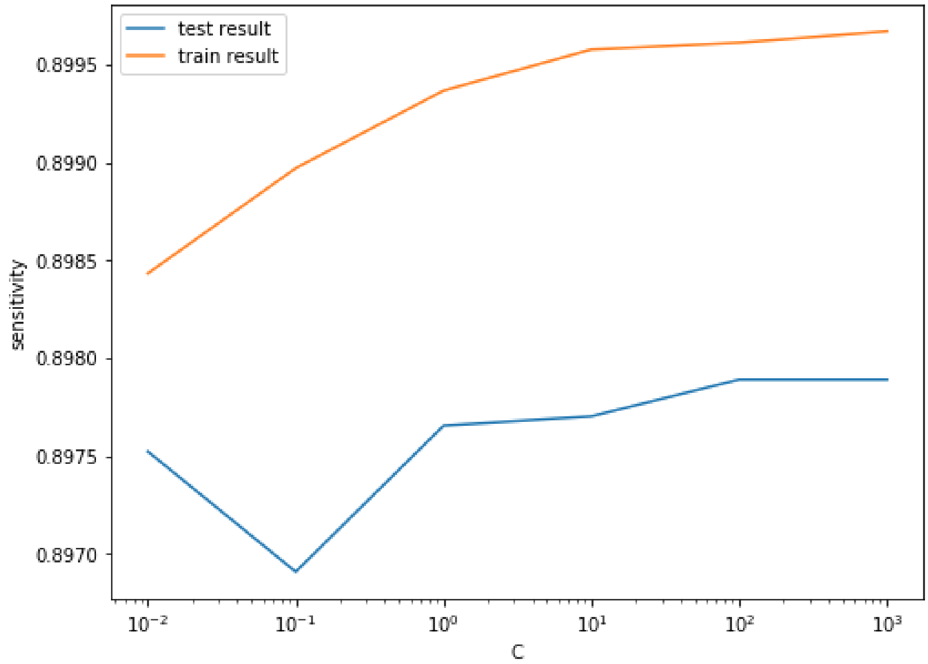



```
# Plotting scree plot
fig = plt.figure(figsize = (10,6))
plt.plot(variance_cumu)
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Variance')
```

```
Text(0, 0.5, 'Cumulative Variance')
```



```python
# plot of C versus train and validation scores

plt.figure(figsize=(8, 6))
plt.plot(cv_results['param_C'], cv_results['mean_test_score'])
plt.plot(cv_results['param_C'], cv_results['mean_train_score'])
plt.xlabel('C')
plt.ylabel('sensitivity')
plt.legend(['test result', 'train result'], loc='upper left')
plt.xscale('log')
```



Random forest with PCA

```
[ ] # Importing random forest classifier
    from sklearn.ensemble import RandomForestClassifier
```

Hyperparameter tuning

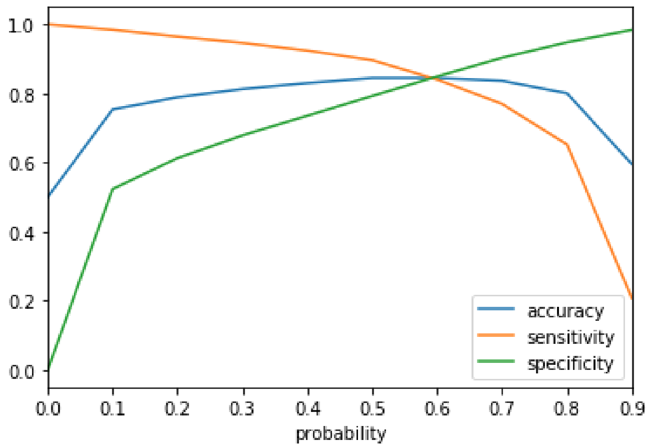
```
param_grid = {
    'max_depth': range(5,10,5),
    'min_samples_leaf': range(50, 150, 50),
    'min_samples_split': range(50, 150, 50),
    'n_estimators': [100,200,300],
    'max_features': [10, 20]
}
# Create a based model
rf = RandomForestClassifier()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf,
                           param_grid = param_grid,
                           cv = 3,
                           n_jobs = -1,
                           verbose = 1,
                           return_train_score=True)

# Fit the model
grid_search.fit(X_train_pca, y_train)
```

Creating a dataframe with the actual churn and the predicted probabilities

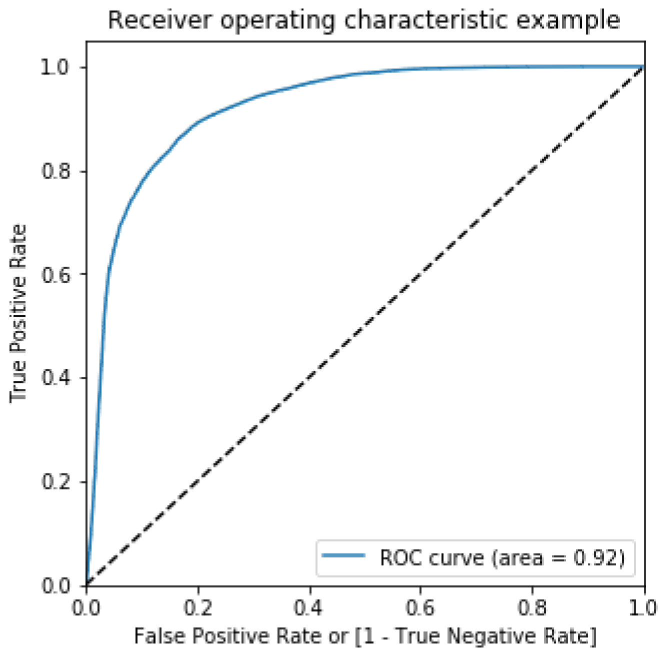
```
y_train_pred_final = pd.DataFrame({'churn':y_train.values, 'churn_prob':y_train_pred_no_pca.values})

#Assigning Customer ID for each record for better readblity
#CustID is the index of each record.
y_train_pred_final['CustID'] = y_train_pred_final.index

y_train_pred_final.head()
```

	churn	churn_prob	CustID
0	0	2.687411e-01	0
1	0	7.047483e-02	1
2	0	8.024370e-02	2
3	0	3.439222e-03	3
4	0	5.253815e-19	4

```
# Plotting accuracy, sensitivity and specificity for different probabilities.
cutoff_df.plot('probability', ['accuracy','sensitivity','specificity'])
plt.show()
```



```
] draw_roc(y_train_pred_final['churn'], y_train_pred_final['churn_prob'])
```




```
y_test_pred_final.head()
```

	churn	CustID	0
0	0	5704	0.034015
1	0	64892	0.000578
2	0	39613	0.513564
3	0	93118	0.020480
4	0	81235	0.034115

```
[ ] # Renaming the '0' column as churn probablity
    y_test_pred_final = y_test_pred_final.rename(columns={0:'churn_prob'})
```

```
[ ] # Rearranging the columns
    y_test_pred_final = y_test_pred_final.reindex_axis(['CustID','churn','churn_prob'], axis=1)
```

Business recomendation

Top predictors

Below are few top variables selected in the logistic regression model.

Variables	Coefficients
loc_ic_mou_8	-3.3287
og_others_7	-2.4711
ic_others_8	-1.5131
isd_og_mou_8	-1.3811
decrease_vbc_action	-1.3293
monthly_3g_8	-1.0943
std_ic_t2f_mou_8	-0.9503
monthly_2g_8	-0.9279
loc_ic_t2f_mou_8	-0.7102
roam_og_mou_8	0.7135

We can see most of the top variables have negative coefficients. That means, the variables are inversely correlated with the churn probablity.

Plots of important predictors for churn and non churn customers

```
# Plotting loc_ic_mou_8 predictor for churn and not churn customers
fig = plt.figure(figsize=(10,6))
sns.distplot(data_churn['loc_ic_mou_8'],label='churn',hist=False)
sns.distplot(data_non_churn['loc_ic_mou_8'],label='not churn',hist=False)
plt.show()
```