Ensambles con usuarios de Twitter

Informe TP2 - Aprendizaje Automático - Grupo 6

Ignacio Chiapella, Juan Knebel, Christian Marcusa

Maestría en Data Mining, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

Entregado el 26 de Julio de 2019

1. Abstract

Con la evolución de las redes sociales, internet, y las tecnologías que se apoyan sobre ambos, cada día es más difícil predecir si la información consumida es real o no. Sobre esta premisa existen además de las “Noticias falsas” (Fake News), perfiles falsos que se dedican a generarla y transmitirla. Estos perfiles pueden ser a su vez personas reales (los hoy llamados “**trolls**”) o robots (“**bots**”), que en muchos casos son programados para re transmitir información y generar tendencia en las redes sociales de **forma automática**. En algunos casos resulta evidente para una persona darse cuenta si la información fue efectivamente escrita por otra persona, o no. Si bien una persona puede clasificar a los usuarios, al tratarse de una cantidad inmanejable se necesitan procesos que puedan “aprender” a estas técnicas, por tal motivo presentaremos dos técnicas usuales de **aprendizaje automático** o **machine learning** que pueden ser utilizadas para clasificar usuarios de redes sociales.

Keywords: Aprendizaje automático, ensambles, robot, bot, publicaciones, trolls, boosting, bagging, stacking

---

2. Introducción

El dominio del problema es crear un producto que pueda detectar si un usuario de la red social **Twitter** es efectivamente un **bot** o una persona real presentado como un un problema de clasificación muy común en los días que corren.

El objetivo de este informe es presentar una descripción de los Ensambles utilizados para la clasificación con distintas estrategias y clasificadores.

El informe se encuentra organizado primero con una breve introducción a los datos (sin repetir mucho ya que se utilizan los datos del trabajo anterior). A continuación se detallan los métodos de aprendizaje automático utilizados y sus respectivos hiperparametros, y luego sus respectivos ensambles y como fueron encarados para el experimento. En otra sección se verán los resultados obtenidos del punto anterior la comparación de de los mismos. Para finalizar daremos las conclusiones que obtuvimos luego de los experimentos y propondremos mejoras para trabajos futuros.

3. Datos

Se utilizó una base de datos obtenida del sitio kaggle que contiene un conjunto de usuarios de la red social Twitter y los mismos se encuentran clasificados en bot o no bot. Los datos originalmente contenían 19 columnas que no todas son necesarias para la clasificación y otras que requieren de otro tipo de análisis (como por ejemplo text mining) que no serán cubiertos en este trabajo práctico. Se realizó un proceso de preprocesamiento de los datos (normalizacion, selección, formato, etc) el cual se explicó en el TP1 y no modificamos para realizar este TP2.

4. Metodología

4.1 Particionado de datos

Antes de comenzar con la experimentación de los métodos se particionó el conjunto de datos, previamente des ordenados de manera aleatoria, en conjunto de **entrenamiento** y otro de **test**, usando una proporción del 70/30. Los datos elegidos para test solamente fueron utilizados para calcular la precisión final de los algoritmos.

Con este criterio se obtuvieron 1938 datos de entrenamiento y 831 seleccionados para medir la efectividad final de los algoritmos.

A su vez al conjunto de datos de entrenamiento a su vez se lo volvió a separar en dos, **entrenamiento** y **validación** pero en este caso con una proporción de 80/20.

Finalmente quedaron 1550 datos de entrenamiento y 388 de validación para hacer los ajustes necesarios.

4.2 Random Forest (Bagging)

En primer lugar se ha implementado un clasificador **Random Forest** para los datos, el cual consiste en **Bagging** con árboles y un ajuste de ganancia para que elija las raíces de formas estocásticas.

Implementamos un grid search en el cual probamos los siguientes parametros:

- Cantidad de árboles a ensamblar
- Máxima profundidad de los árboles
- Ganancia: entropia o Gini Gain
- Máxima cantidad de atributos a utilizar en cada nivel del árbol
- Si utiliza mecanismo de bootstraping o no

A continuación mostramos los mejores clasificadores que se obtuveron.

Sin la utilización de K - Cross Validation

RandomForestClassifier(bootstrap=False, class_weight=None, criterion='entropy',
max_depth=None, max_features='sqrt', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=50,
n_jobs=None, oob_score=False, random_state=None,
verbose=0, warm_start=False)

Con la utilización de K - Cross Validation con K = 5

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
max_depth=None, max_features='sqrt', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100,
n_jobs=None, oob_score=False, random_state=None,
verbose=0, warm_start=False)

4.3 Mezcla de clasificadores con votación (Stacking)

Luego se ha hecho una prueba con un ensamble que se ha generado es un stack con 3 clasificadores, un **Árbol de Decisión**, una **Regresión Logística** y un **KNN** (Vecinos más cercanos). A su salida se implementa una **Votación** para reducir el sesgo. Para el estimador de árbol de decisión y KNN también utilizamos un grid search, en el caso de del árbol de decisión se probó:

- Máxima profundidad
- Máxima cantidad de hojas
- Ganancia: entropia o Gini Gain

Y en el caso de KNN:

- Cantidad de vecinos
- Cálculo de los pesos en las predicciones
- Algoritmo utilizado para el cómputo de vecinos

A continuación mostramos los mejores clasificadores que se obtuveron.

Sin la utilización de K - Cross Validation

DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')

KNeighborsClassifier(algorithm='ball_tree', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=14, p=2,
weights='distance')

Con la utilización de K - Cross Validation con K = 5

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=6,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')

KNeighborsClassifier(algorithm='ball_tree', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=7, p=2,
weights='uniform')

4.4 ADABoost (Boosting)

Se ha utilizado el clasificador de **ADABoost**, que por defecto utiliza Árboles de decisión, y se ha utilizado este para comparar tanto con **Random Forest** como con el stack. En particular utilizamos el árbol de decisión que obtuvimos en el método de Stacking.

5. Resultados

Sin la utilización de K - Cross Validation

Método	Accuracy	TN	FP	FN	Tp	Precisión	Recall	F 1
Random Forest	0.89	402	43	50	340	0.88	0.87	0.87
Decision Tree	0.85	376	60	65	325	0.844	0.83	0.83
KNN	0.69	304	142	113	277	0.66	0.71	0.68
LogisticRegression	0.73	239	198	20	370	0.65	0.95	0.77
Ensamble	0.79	309	138	38	352	0.71	0.90	0.80
AdaBoost	0.86	389	43	76	314	0.87	0.80	0.84

Con la utilización de K - Cross Validation con K = 5

Método	Accuracy	TN	FP	FN	Tp	Precisión	Recall	F 1
Random Forest	0.89	407	34	56	334	0.90	0.85	0.88
Decision Tree	0.84	397	44	83	307	0.87	0.78	0.82
KNN	0.70	309	132	110	280	0.67	0.71	0.69
LogisticRegression	0.73	239	202	21	369	0.64	0.94	0.76
Ensamble	0.79	319	122	46	344	0.73	0.88	0.80
AdaBoost	0.87	402	39	61	329	0.89	0.84	0.86

Resumen de Scores					
	knn	dt	log_reg	rf	ada
SIN CV	0.7148	0.8929	0.7316	0.8929	0.8712
CON CV	0.7087	0.8892	0.7316	0.8892	0.8832

6. Conclusiones

- Se observa que el Random Forest (Bagging) arroja mejores métricas a la hora de clasificar el conjunto de Test. Seguido por el Boosting, y en tercer lugar por el DecisionTree. El Stacking tiene muy alto Recall, ya que la Regresión Logística tiene una tasa de Recall muy elevada respecto a su Precisión (Y al recall de los árboles).
- Se observa que para la problemática en cuestión, el árbol de decisión resultó ser el mejor clasificador sin ensamblar (P: 0.84 R: 0.83), y luego podemos ensamblar de distintas maneras depende si queremos mejorar la Precisión o el Recall.
- Cuando realizamos el set de pruebas con y sin cross validation, lo que observamos es que en todos los casos sin CV dan mejor salvo en el caso del ADA.

7. Trabajos futuros

Se pueden implementar estos ensambles con distintos algoritmos de clasificación. Si bien en este problema la clasificación por árboles resulta muy buena, se puede entrenar una red multicapa o directamente ensamblar otros algoritmos que ajustan mejor al problema.

8. Referencias bibliográficas

Link al Dataset

https://www.kaggle.com/charvijain27/detecting-twitter-bot-data