Análisis de una red social.
Estructuras Modulares en Redes Complejas

Mónica Rampérez Andrés

Abril, 2021

Resumen

En este documento se refleja un estudio sobre la red social *Epinions*, una antigua plataforma web en la que cada persona podía dar sus opiniones sobre diversos productos. Para este trabajo se han utilizado los datos de su sistema de reputación, generando un grafo del cual se han extraído características para cada una de las relaciones de confianza o desconfianza existentes entre los usuarios. Posteriormente, se han entrenado distintos tipos de clasificadores con esta información y se ha realizado un experimento para seleccionar aquellos que son significativamente mejores.

1. Introducción

Como hemos visto a lo largo del curso, la **teoría de grafos** [1] tiene aplicaciones directas a la hora de resolver problemas en el mundo real, y una de las más interesantes en la actualidad tiene que ver con las redes sociales.

Desde el punto de vista de la teoría de grafos, una **red social** [2] se puede representar como un conjunto de nodos correspondientes a cada uno de los **individuos** pertenecientes a ella y un conjunto de arcos que ilustrarían aquellas **relaciones** existentes entre los mismos. Estudiar las redes sociales desde este enfoque permite extraer información muy interesante e incluso medir determinadas propiedades y estudiar aquellas semejanzas o diferencias que puedan existir entre grupos de individuos.

Esta disciplina se conoce como **análisis de redes sociales** [3], y nace motivada por aportaciones de diferentes ciencias como las matemáticas, la psicología o la sociología. A partir de todas ellas surge la pregunta o el interés de comprender como funcionan los fenómenos relacionales entre grandes cantidades de individuos.

Especialmente en esta última década, el interés por este tipo de análisis ha aumentado considerablemente, haciéndose notar su gran utilidad en aplicaciones para el mundo real, en las que se incluyen desde la persecución de redes criminales hasta la transmisión de enfermedades infecciosas, incluyendo el modelado de veredictos en procesos judiciales [4].

En el caso concreto de este proyecto, la red social analizada será la constituida por el **sistema de reputación** [5] del portal web *Epinions* [6], basado en la confianza o desconfianza entre sus diversos usuarios. Una vez recolectados los datos, se trabajará con el grafo estructurado con ellos para extraer un vector de características correspondiente a cada una de las relaciones presentes, que finalmente serán utilizados en su conjunto para entrenar diversos tipos de clasificadores. Por último, se realizarán algunos test estadísticos para contrastar los resultados obtenidos en cuanto a sus métricas de rendimiento.

Cabe destacar que para el análisis de redes sociales es imprescindible el uso de **herramientas matemáticas e informáticas** que permitan trabajar con la información disponible en este tipo de datos. En el caso de este estudio, se ha utilizado el lenguaje de programación `Python3` y algunas de sus librerías más conocidas, como `igraph` [7] para el manejo de grafos o `sklearn` [8] para el uso de clasificadores.

2. Dataset utilizado

2.1. *Epinions*

El dataset seleccionado para realizar el estudio es el correspondiente a *Epinions*, un antiguo portal web en el que los usuarios podían leer diversas críticas sobre artículos de todo tipo, como software, música o programas de televisión. Además, los usuarios podían recibir ingresos en función de lo útiles que eran sus *reviews*. Por este mismo motivo, se incorporó un **sistema de reputación** [5], mediante el cual cada individuo podía indicar los usuarios en los que confiaba y aquellos en los que no, incrementando la calidad y seguridad del portal.

Figura 1: Logotipo de *Epinions.com*

Para el desarrollo de esta práctica se ha utilizado como base de datos inicial aquella información disponible correspondiente a este sistema de reputación, partiendo de todas las puntuaciones positivas (`+1`) o negativas (`-1`) que se han dado entre cada par de usuarios `A->B` (disponible como `user_rating.txt`).

2.2. Preprocesado

La base de datos original cuenta con la información de 841372 puntuaciones dadas de un usuario `A` a un usuario `B`, con su correspondiente *rating* positivo (`+1`) o negativo (`-1`) y la fecha de aplicación.

Una vez leídos los datos, la primera etapa de su preprocesado ha consistido en disminuir el número de nodos (usuarios) para que sea más manejable y menos costoso computacionalmente. En este caso se ha hecho una reducción a **500 nodos**, quedando un total de **1654 relaciones** entre ellos.

Una vez preparados los datos de esta manera, se ha confeccionado la **matriz de adyacencia** (disponible como `adj_matrix.csv`) correspondiente al **grafo dirigido y ponderado** que representa. La matriz resultante es una matriz A de tamaño 500x500, para la que cada elemento A_{ij} representa la puntuación dada por el usuario i al usuario j, pudiendo ser `+1` o `-1` en caso de existir dicha valoración, o `0` en caso no haberla.

3. Extracción de características

A partir de la matriz de adyacencia A, es posible generar una estructura de grafo utilizando la librería `igraph`. Esta librería incluye muchas funcionalidades para el estudio de grafos, permitiendo calcular de manera sencilla las medidas que se utilizarán para extraer las características [7].

A la hora de construir la matriz con las características (disponible como `feature_matrix.csv`), cada fila de la matriz contendrá la información correspondiente a cada relación `A->B` existente. Para extraer las características de cada una de estas relaciones, primero se ha eliminado dicha arista del grafo, después se han calculado sus medidas y finalmente se ha vuelto a incorporar.

3.1. Características utilizadas

Las medidas utilizadas para extraer las características de cada relación `A->B` quedan reflejadas en la Tabla 1.

Las 4 primeras medidas utilizadas corresponden a valores generales para el grafo, mientras que las 9 siguientes son valores que se calculan a nivel de nodo, por lo que para cada una de ellas se han extraído dos medidas, la del nodo que puntúa y la del nodo que recibe la puntuación, obteniendo así un total de 22 *features*.

Cabe destacar que es posible encontrar una definición más extensa y la explicación exacta del cálculo de cada una de las medidas en la documentación oficial de la librería `igraph` [7].

Propiedad	Nombre	Definición
Average Path Length	`AvgPathLen`	Distancia mínima media entre cada par de nodos
Assortativity	`Assort`	Nivel de homofilia del grafo
Transitivity	`Transit`	Probabilidad de que los nodos adyacentes a un nodo estén conectados entre sí
Reciprocity	`Recipr`	Proporción de conexiones mutuas
Strength	`StrengthA` `StrengthB`	Suma de los pesos de las aristas adyacentes a cada nodo
Pagerank	`PagerankA`	Puntuación PageRank de Google para cada nodo
	`PagerankB`	
Harmonic Centrality	`HarmCenA` `HarmCenB`	Media del inverso de la distancia de un nodo al resto
Eigenvector Centrality	`EigenCenA`	Conectividad que posee cada nodo dentro del grafo
	`EigenCenB`	
Eccentricity	`EccentrA`	Distancia de la ruta más corta de cada nodo al más lejano a él
	`EccentrB`	
Coreness	`CorenessA`	Mayor orden k del k-core que contiene cada nodo
	`CorenessB`	
Betweenness	`BetweennA` `BetweennB`	Número de caminos (más cortos) que pasan por cada nodo
Hub Score	`HubScrA`	Principal autovector de A*t(A), siendo A la matriz de adyacencia del grafo
	`HubScrB`	
Authority Score	`AuthScrA` `AuthScrB`	Principal autovector de t(A)*A, siendo A la matriz de adyacencia del grafo

Tabla 1: Medidas utilizadas para extraer las características de cada relación

3.2. *Oversampling*

Una vez generado el vector de características de cada relación existente, es necesario comprobar como se encuentran distribuidas las clases. En este caso, existe un claro **desbalanceo**, ya que entorno a un `85%` de las valoraciones son positivas mientras que el `15%` son negativas.

Este fenómeno podría perjudicar las posteriores clasificaciones, por lo que se ha decidido realizar un **sobremuestreo** o *oversampling* para la clase minoritaria, utilizando la técnica `SMOTE` [9] proporcionada por la librería `imblearn`. De esta manera se ha conseguido finalmente una matriz de características para 1472 valoraciones positivas y 1472 valoraciones negativas, quedando el dataset **balanceado**.

Todo el código correspondiente hasta este punto del trabajo se encuentra en el fichero `sna_feat.py`.

4. Experimentos propuestos

4.1. Clasificadores utilizados

Una vez preparada la matriz con los datos necesarios, el objetivo es **entrenar varios clasificadores para contrastar su rendimiento**. Cabe destacar que antes de realizar el entrenamiento se ha realizado un *tuning* de los parámetros para cada uno de ellos, de manera que los resultados obtenidos sean los mejores posibles. En este caso, los clasificadores escogidos de la librería `sklearn` [8] son los siguientes:

- **Naive Bayes**. `GaussianNB()`, con *var_smoothing* = 0.0.
  Esta técnica de clasificación supervisada es una de las más utilizadas por su simplicidad y rapidez. Se basa en la construcción de modelos para predecir la probabilidad de los posibles resultados. La propia librería dispone de diversas versiones de este clasificador, pero tras hacer una pequeña prueba, el clasificador gaussiano era el que mejores resultados mostraba.

- **k-Nearest Neighbors**. `KNeighborsClassifier()`, con *n_neighbors* = 1 y *metric* = Minkowski.
  Otro de los clasificadores más utilizados por su gran rapidez y sencillez. En este caso se basa en el cálculo de la distancia de Minkowski al vecino más cercano.

- **Decision Tree**. `DecisionTreeClassifier()`, con *criterion* = entropy y *max_depth* = 30.
  Otro clasificador muy conocido, que toma las decisiones siguiendo una estructura de árbol, donde las hojas representarían las etiquetas de clase mientras que las ramas representarían las conjunciones de características que llevarían a cada una de ellas.

- **Random Forest**. `RandomForestClassifier()`, con *max_depth* = 18 y *n_estimators* = 550.
  Basado en el clasificador anterior de árbol de decisión, consiste en una combinación de los mismos, generando multitud de árboles no correlacionados que luego se promedian.

- **Red Neuronal - MLP**. `MLPClassifier()`, con *activation* = tanh, *alpha* = 0.0001, *hidden_layer_sizes* = (50, 100, 50), *learning_rate* = adaptive y *solver* = adam.
  La red neuronal seleccionada es un perceptrón multicapa (MLP) con la estructura indicada en la Figura 2. Este clasificador tiene la ventaja de ser capaz de resolver problemas que no son linealmente separables.

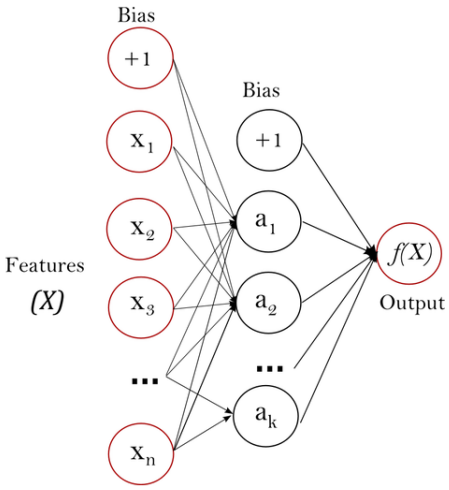

Figura 2: Esquema del perceptrón multicapa (de la documentación de `sklearn`)

Estos clasificadores se han aplicado sobre la matriz de datos utilizando una técnica de **10-fold cross validation** [10]. Esto supone dividir los datos de muestra en 10 subconjuntos, de manera que en cada una de las 10 iteraciones se usa uno de ellos como conjunto de test y el resto como conjunto de entrenamiento. Finalmente, se puede calcular la media aritmética de los resultados para obtener un único valor.

4.2. Métricas de rendimiento

Para cada uno de los clasificadores se han valorado distintas **medidas de rendimiento** [11], basadas en la matriz de confusión obtenida en cada ejecución. Esta matriz nos indica el número de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN) en la clasificación.

Concretamente, las 5 métricas utilizadas han sido las siguientes:

- **Accuracy**. También denominada como exactitud, refleja el porcentaje de casos que el modelo ha clasificado correctamente. Se calcula como:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**. La precisión de un clasificador se puede interpretar como lo cerca que está el resultado de una predicción del valor verdadero. Se calcula como:

$$precision = \frac{TP}{TP + FP}$$

- **Recall**. La exhaustividad del clasificador se refiere a la tasa de verdaderos positivos que es capaz de identificar. Se calcula como:

$$recall = \frac{TP}{TP + FN}$$

- **F-Score**. Este valor se utiliza para combinar las medidas de `precision` y `recall` en un solo valor. Se calcula como:

$$Fscore = 2 \cdot \frac{precison \cdot recall}{precison + recall}$$

- **Área bajo la curva ROC**. Esta medida nos da una idea global del buen funcionamiento del clasificador.

  Cabe destacar que no se ha usado la métrica MCC ya que tiene más utilidad en escenarios no balanceados.

5. Resultados

Una vez ejecutado el experimento, se ha obtenido para cada clasificador un vector de 10 mediciones de cada una de las métricas comentadas anteriormente.

Estos resultados quedan reflejados en las siguientes tablas:

Naive Bayes (NB)

	0	1	2	3	4	5	6	7	8	9	**mean**
accuracy	0.67	0.82	0.75	0.73	0.71	0.75	0.54	0.61	0.59	0.6	**0.68**
precision	0.64	0.76	0.74	0.73	0.7	0.74	0.56	0.63	0.59	0.62	**0.67**
recall	0.78	0.93	0.77	0.74	0.73	0.76	0.41	0.55	0.58	0.51	**0.68**
f-score	0.7	0.84	0.75	0.73	0.71	0.75	0.47	0.59	0.58	0.56	**0.67**
roc_auc	0.76	0.87	0.82	0.8	0.76	0.8	0.49	0.69	0.64	0.6	**0.72**

Tabla 2: Métricas obtenidas con el clasificador Naive Bayes

k-Nearest Neighbors (kNN)

	0	1	2	3	4	5	6	7	8	9	**mean**
accuracy	0.86	0.83	0.84	0.85	0.82	0.88	0.79	0.88	0.82	0.82	**0.84**
precision	0.84	0.84	0.88	0.89	0.89	0.88	0.86	0.89	0.85	0.86	**0.87**
recall	0.89	0.83	0.78	0.81	0.73	0.88	0.69	0.86	0.78	0.77	**0.8**
f-score	0.87	0.83	0.83	0.85	0.8	0.88	0.76	0.88	0.82	0.81	**0.83**
roc_auc	0.86	0.83	0.84	0.85	0.82	0.88	0.79	0.88	0.82	0.82	**0.84**

Tabla 3: Métricas obtenidas con el clasificador kNN

Decision Tree (DT)

	0	1	2	3	4	5	6	7	8	9	**mean**
accuracy	0.75	0.89	0.96	0.93	0.9	0.9	0.91	0.93	0.89	0.82	**0.89**
precision	0.68	0.88	0.99	0.96	0.97	0.93	0.94	0.96	0.91	0.91	**0.91**
recall	0.94	0.93	0.86	0.9	0.8	0.84	0.83	0.89	0.88	0.74	**0.86**
f-score	0.77	0.91	0.94	0.92	0.9	0.9	0.89	0.93	0.89	0.8	**0.88**
roc_auc	0.77	0.93	0.93	0.94	0.88	0.89	0.91	0.92	0.9	0.82	**0.89**

Tabla 4: Métricas obtenidas con el clasificador Decision Tree

Random Forest (RF)

	0	1	2	3	4	5	6	7	8	9	**mean**
accuracy	0.83	0.96	0.98	0.97	0.95	0.98	0.91	0.97	0.95	0.9	**0.94**
precision	0.77	0.96	1.0	0.98	1.0	0.97	0.98	0.99	0.99	0.98	**0.96**
recall	0.95	0.97	0.95	0.95	0.89	0.99	0.85	0.95	0.9	0.82	**0.92**
f-score	0.86	0.97	0.98	0.96	0.95	0.98	0.91	0.97	0.94	0.88	**0.94**
roc_auc	0.95	0.99	1.0	0.99	1.0	1.0	0.99	1.0	0.99	0.96	**0.99**

Tabla 5: Métricas obtenidas con el clasificador Random Forest

Red Neuronal - MLP (MLP)

	0	1	2	3	4	5	6	7	8	9	**mean**
accuracy	0.75	0.77	0.75	0.72	0.68	0.68	0.54	0.63	0.59	0.6	**0.67**
precision	0.65	0.77	0.75	0.71	0.64	0.71	0.57	0.6	0.61	0.65	**0.67**
recall	0.84	0.85	0.55	0.48	0.73	0.69	0.33	0.59	0.6	0.56	**0.62**
f-score	0.79	0.79	0.68	0.73	0.71	0.69	0.47	0.6	0.62	0.55	**0.66**
roc_auc	0.9	0.9	0.79	0.75	0.74	0.77	0.55	0.68	0.68	0.65	**0.74**

Tabla 6: Métricas obtenidas con el clasificador Red Neuronal MLP

En principio, los clasificadores Naive Bayes (Tabla 2) y la Red Neuronal MLP (Tabla 6) parecen tener un peor rendimiento. Los resultados de k-Nearest Neighbors (Tabla 3) y Decision Tree (Tabla 4) muestran mejores resultados, tan solo superados por el clasificador Random Forest (Tabla 5).

6. Significatividad de los resultados

A priori, observando las tablas, se podría inferir que el clasificador Random Forest realiza una mejor clasificación que el resto. Sin embargo, para comparar los resultados de manera rigurosa y asegurar que un clasificador es mejor que los otros, es necesario utilizar **tests estadísticos** [12].

En este caso se va a comprobar si para cada una de las métricas hay diferencias significativas entre los 5 clasificadores entrenados, utilizando las funcionalidades proporcionadas por la librería `scipy`.

Para ello, para cada métrica en cada par de clasificadores, primero se realiza un test `Shapiro Wilk` [13], que propone como hipótesis nula que los datos siguen una distribución normal. Si el *p-valor* de este test resulta superior al umbral $\alpha = 0{,}05$, se asumirá normalidad en los datos y se realizará un test `ANOVA` [14]. En el caso de no poder asumir normalidad se realizará el test `Kruskal Wallis` [14]. Ambos tests establecen como hipótesis nula que no existe diferencia significativa entre los clasificadores, lo que se confirmará si su *p-valor* se mantiene por encima del umbral $\alpha = 0{,}05$.

Las siguientes tablas indican para cada pareja de clasificadores cual de ellos funciona significativamente mejor tras realizar los tests correspondientes.

6.1. Accuracy

La Tabla 7 muestra los mejores clasificadores para la métrica `accuracy`. En este caso Random Forest es significativamente mejor que el resto.

accuracy

	NB	**kNN**	**DT**	**RF**	**MLP**
NB		kNN	DT	RF	=
kNN	kNN		DT	RF	kNN
DT	DT	DT		RF	DT
RF	RF	RF	RF		RF
MLP	=	kNN	DT	RF	

Tabla 7: Comparativa de *accuracy* entre clasificadores

6.2. Precision

La Tabla 8 muestra los mejores clasificadores para la métrica `precision`. En este caso Random Forest también es significativamente mejor que el resto.

precision

	NB	**kNN**	**DT**	**RF**	**MLP**
NB		kNN	DT	RF	=
kNN	kNN		DT	RF	kNN
DT	DT	DT		RF	DT
RF	RF	RF	RF		RF
MLP	=	kNN	DT	RF	

Tabla 8: Comparativa de *precision* entre clasificadores

6.3. Recall

La Tabla 9 muestra los mejores clasificadores para la métrica `recall`. En este caso Random Forest también es significativamente mejor que el resto.

recall

	NB	**kNN**	**DT**	**RF**	**MLP**
NB		kNN	DT	RF	=
kNN	kNN		=	RF	kNN
DT	DT	=		RF	DT
RF	RF	RF	RF		RF
MLP	=	kNN	DT	RF	

Tabla 9: Comparativa de *recall* entre clasificadores

6.4. F-score

La Tabla 10 muestra los mejores clasificadores para la métrica `f-score`. En este caso Random Forest también es significativamente mejor que el resto.

f-score

	NB	**kNN**	**DT**	**RF**	**MLP**
NB		kNN	DT	RF	=
kNN	kNN		DT	RF	kNN
DT	DT	DT		RF	DT
RF	RF	RF	RF		RF
MLP	=	kNN	DT	RF	

Tabla 10: Comparativa de *f-score* entre clasificadores

6.5. Área bajo la curva ROC

La Tabla 11 muestra los mejores clasificadores para la métrica `roc_auc`. En este caso Random Forest también es significativamente mejor que el resto.

roc_auc

	NB	**kNN**	**DT**	**RF**	**MLP**
NB		kNN	DT	RF	=
kNN	kNN		DT	RF	kNN
DT	DT	DT		RF	DT
RF	RF	RF	RF		RF
MLP	=	kNN	DT	RF	

Tabla 11: Comparativa del área bajo la curva ROC entre clasificadores

Gracias a los tests estadísticos, podemos confirmar que Random Forest es el mejor clasificador, seguido de Decision Tree y k-Nearest Neighbors. En el caso de Naive Bayes y la Red Neuronal - MLP no existen diferencias significativas entre ellos, pero tienen un rendimiento peor que el resto.

Todo el código correspondiente a esta parte del trabajo se encuentra en el fichero `sna_class.py`.

7. Conclusiones

A lo largo de este trabajo se ha pretendido ilustrar la gran utilidad que puede llegar a tener la teoría de grafos y el análisis de datos en general dentro de un área de estudio con tanta relevancia en la actualidad como es el de las redes sociales.

En este caso, se ha tomado como red el conjunto de usuarios del portal web *Epinions*, para el que se conocen las relaciones de confianza y desconfianza existentes entre ellos. Interpretar esta información tras representarla como una estructura de grafo ha permitido que sea posible hacer uso de la teoría de grafos para calcular diversas medidas y propiedades del mismo. De esta manera se ha conseguido extraer características para cada una de las relaciones existentes haciendo uso de la potente librería `igraph`.

El uso de estas características para entrenar clasificadores, y los propios experimentos realizados gracias a `sklearn`, han mostrado que es posible alcanzar resultados realmente buenos como en el caso del **clasificador Random Forest**, con un accuracy de en torno al `94%` o incluso una precisión del `96%`. Además, estas comparativas han sido rigurosamente confirmadas mediante el uso de tests estadísticos como `Shapiro Wilk`, `ANOVA` y `Kruskal Wallis`.

Esto supondría que, conociendo las características de cualquier par de nodos en la red, sería posible saber con bastante fiabilidad si existiría una relación de confianza o desconfianza entre ellos. Este tipo de información puede llegar a ser de gran utilidad en muchos escenarios, como por ejemplo a la hora de desarrollar aplicaciones basadas en sistemas de recomendación o para la creación de perfiles de usuario.

Cabe destacar que, a pesar de que en los experimentos el clasificador Random Forest se ha confirmado como el mejor de ellos, sería interesante como trabajo futuro probar otro tipo de clasificadores o incluso utilizar otras métricas para comparar el rendimiento entre ellos. Por ejemplo, en este caso no se ha tenido en cuenta el tiempo que tarda cada uno de ellos en realizar la clasificación, y es posible que en algunas aplicaciones sea necesario tenerlo en cuenta para alcanzar un *trade-off* adecuado entre la obtención de buenos resultados y su coste computacional.

De la misma manera, también sería interesante hacer un estudio más exhaustivo de las medidas utilizadas para la extracción de características, probando otras distintas o incluso realizando alguna selección o análisis de componentes principales.

En definitiva, el análisis de redes sociales desde el punto de vista de la teoría de grafos es un área de estudio realmente interesante y con mucho potencial en la actualidad, para el que ya existe software y librerías muy potentes como las de `Python` que permiten sacarle el máximo partido.

Referencias

[1] Bondy, J. A., & Murty, U. S. R. (1976). Graph theory with applications (Vol. 290). London: Macmillan.

[2] Liu, W., Sidhu, A., Beacom, A. M., & Valente, T. W. (2017). Social network theory. The international encyclopedia of media effects, 1-12.

[3] Wasserman, S., & Faust, K. (1994). Social network analysis: Methods and applications.

[4] Masías, V. H., Valle, M., Morselli, C., Crespo, F., Vargas, A., & Laengle, S. (2016). Modeling verdict outcomes using social network measures: the watergate and caviar network cases. PloS one, 11(1), e0147248.

[5] Chow, W. S., & Chan, L. S. (2008). Social network, social trust and shared goals in organizational knowledge sharing. Information & management, 45(7), 458-465.

[6] Massa, P., & Avesani, P. (2005, July). Controversial users demand local trust metrics: An experimental study on epinions.com community. In AAAI (Vol. 5, pp. 121-126).

[7] Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. InterJournal, complex systems, 1695(5), 1-9.

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[9] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[10] Fushiki, T. (2011). Estimation of prediction error by using K-fold cross-validation. Statistics and Computing, 21(2), 137-146.

[11] Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006, December). Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In Australasian joint conference on artificial intelligence (pp. 1015-1021). Springer, Berlin, Heidelberg.

[12] Lehmann, E. L., & Romano, J. P. (2006). Testing statistical hypotheses. Springer Science & Business Media.

[13] Hanusz, Z., Tarasinska, J., & Zielinski, W. (2016). Shapiro-Wilk test with known mean. REVSTAT-Statistical Journal, 14(1), 89-100.

[14] Hecke, T. V. (2012). Power study of anova versus Kruskal-Wallis test. Journal of Statistics and Management Systems, 15(2-3), 241-247.