UNIVERSIDAD POLITÉCNICA DE VICTORIA

A METHODOLOGY FOR COMPARING BINARY CLASSIFIERS WITH CV AND ROC CURVES

PROTOCOLO DE INVESTIGACIÓN
DE LA CARRERA
INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PRESENTAN:
ARMANDO ISAAC HERNANDEZ MUÑIZ 1530343
IRLANDA ISAMAR TORRES MORENO 1430238
AARON LEONARDO SÁNCHEZ MARTÍNEZ 1530509

Titular de Mineria de Datos Aplicada
DR. ALBERTO GARCÍA ROBLEDO

CIUDAD VICTORIA, TAMAULIPAS, ABRIL DE 2018

Índice

1 Resumen

Los algoritmos de clasificación permiten validar, crear o probar modelos de clasificación, además de poder crear una asociación entre grupos, que compartan similitudes, además de crear un emparejamiento entre etiquetas de valores, por medio de la precisión en la proceso de la predicción en el cual se toma importancia a variables distintas, como son las continuas, a diferencia de la clasificación, además de la eficiencia en cuanto a los costos computacionales, esto también, de la mano de la robustez que se tiene para trabajar con ruido y soportar la ausencia de valores. Como parte de los métodos de clasificación, también se tiene en cuenta la escalabilidad para trabajar con grandes cantidades de datos, dado el caso de trabajar con datasets en el presente trabajo, por lo cual en el siguiente reporte se presenta una comparación de algoritmos que permiten poner a prueba el dataset "Hayes Roth" con la implementación de la validación cruzada y curvas ROC y los algoritmos de Random Forest, Máquina de Soporte Vectorial y Naive Bayes.

2 Introduccion

2.1 Motivacion

La clasificación en minería de datos permite la identificación de información, además implica enseñarle a un ordenador a clasificar objetos mediante una hipótesis definida, donde las clases son conocidas. La clasificación de datos implica el análisis del funcionamiento y la forma de comportarse de los algoritmos de clasificación. De manera general, la clasificación tiene como objetivo asignar una categoría a un objeto en base de sus otras características, esto para permitir crear modelos predictivos, los cuales puedan describir clases de datos, esto para crear divisiones entre los conjuntos de datos de una clase para así facilitar su comprensión. De esta manera se ponen a prueba las técnicas de validación cruzada y las curvas ROC las cuales son usadas para evaluar los resultados de un análisis estadístico y de esta manera asegurar que son distintos entre los datos de entrenamiento y prueba.

2.2 Descripcion del DataSet

Número de instancias: 132 instancias de entrenamiento, 28 instancias de prueba Número de atributos: 5 más el atributo de membresía de clase. 3 conceptos. Información de atributos:

1. nombre: distinto para cada instancia y representado numéricamente
2. hobby: valores nominales que oscilan entre 1 y 3
3. edad: valores nominales que oscilan entre 1 y 4
4. nivel educativo: valores nominales que oscilan entre 1 y 4
5. estado civil: valores nominales que oscilan entre 1 y 4
6. clase: valor nominal entre 1 y 3

Valores de atributo faltantes: ninguno Descripción detallada del experimento: 1. 3 categorías (1, 2 y ninguna - a la que llamo 3)
- algunas de las instancias pueden clasificarse en clase 1 o 2, y
se han distribuido de manera uniforme entre las dos clases
2. 5 Atributos
- A. nombre (un número generado aleatoriamente entre 1 y 132)
- B. hobby (un número generado aleatoriamente entre 1 y 3)
- C. edad (un número entre 1 y 4)
- D. nivel de educación (un número entre 1 y 4)
- E. estado civil (un número entre 1 y 4)
3. Clasificación:
- solo los atributos C-E son diagnósticos; los valores para A y B son ignorados
- Clase Ni: si se produce un 4 para cualquier atributo C-E
- Clase 1: De lo contrario, si (# de 1's)> (# de 2's) para los atributos C-E
- Clase 2: De lo contrario, if (# of 2's)> (# of 1's) para los atributos C-E
- Ya sea 1 o 2: De lo contrario, if (# of 2's) = (# of 1's) para los atributos C-E
4. Prototipos:
- Clase 1: 111
- Clase 2: 222
- Clase O bien: 333
- Clase Ni: 444
5. Número de instancias de entrenamiento: 132
- Cada instancia presentada 0, 1 o 10 veces
- Ninguno de los prototipos vistos durante el entrenamiento
- Se repiten 3 instancias de cada una de las categorías 1, 2 y cualquiera 10 veces cada
- 3 instancias adicionales de la categoría Cualquiera se muestran durante aprendizaje
5. Número de instancias de prueba: 28
- Todos los 9 clase 1
- Todos 9 clase 2
- Todas las 6 clases
- Los 4 prototipos
– – – – – – – – – –
- 28 total

2.3 Objetivo

El objetivo de esta práctica se resume a comparar de forma metodológica clasificadores binarios aplicados en conjuntos de datos categóricos combinando el uso de curvas ROC y validación cruzada con Python y Scikit-learn.

3 Marco Teórico

3.1 Python

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.[1]

3.2 Jupyter

El Jupyter Notebook es una aplicación web de código abierto que le permite crear y compartir documentos que contengan códigos, ecuaciones, visualizaciones y textos narrativos en vivo. Los usos incluyen: limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos, aprendizaje automático y mucho más.[2]

3.3 Pandas

Pandas es una librería de python destinada al análisis de datos, que proporciona unas estructuras de datos flexibles y que permiten trabajar con ellos de forma muy eficiente. Pandas ofrece las siguientes estructuras de datos: [3]

- Series: Son arrays unidimensionales con indexación (arrays con índice o etiquetados), similar a los diccionarios. Pueden generarse a partir de diccionarios o de listas.
- DataFrame: Son estructuras de datos similares a las tablas de bases de datos relacionales como SQL.
- Panel, Panel4D y PanelND: Estas estructuras de datos permiten trabajar con más de dos dimensiones. Dado que es algo complejo y poco utilizado trabajar con arrays de más de dos dimensiones no trataremos los paneles en estos tutoriales de introdución a Pandas.

3.4 Scikit-Learn

Scikit-learn es una biblioteca de aprendizaje de máquina de software libre para el lenguaje de programación Python. Presenta varios algoritmos de clasificación, regresión y agrupación, incluyendo máquinas de vectores de soporte, bosques aleatorios, aumento de gradiente, k- medias y DBSCAN, y está diseñado para interoperar con las bibliotecas numéricas y científicas de Python, NumPy y SciPy.[4]

3.5 Descripción de los Algoritmos de Clasificación

3.5.1 Naive Bayes

El algoritmo de clasificación Naive Bayes es un clasificador probabilístico. Se basa en modelos de probabilidades que incorporan fuertes suposiciones de independencia.

Es posible derivar modelos de probabilidades utilizando el teorema de Bayes (atribuido a Thomas Bayes). En función de la naturaleza del modelo de probabilidades, el algoritmo Naive Bayes puede prepararse en un entorno de aprendizaje supervisado.
La minería de datos de InfoSphere Warehouse se basa en la máxima probabilidad para el cálculo de parámetros para los modelos Naive Bayes. El modelo Naive Bayes generado se ajusta al estándar PMML (Predictive Model Markup Language).
Un modelo Naive Bayes está formado por un cubo muy grande que incluye las dimensiones siguientes:[5]

- Nombre del campo de entrada
  El valor del campo de entrada para los campos discretos o el rango de valores del campo de entrada para los campos continuos.
- El algoritmo Naive Bayes divide los campos continuos en valores discretos. Valor de campo de destino.

Esto significa que un modelo Naive Bayes registra la frecuencia con la que un valor del campo de destino aparece junto a un valor de un campo de entrada.
El algoritmo de clasificación Naive Bayes incluye el parámetro de umbral de probabilidad ZeroProba. El valor del parámetro de umbral de probabilidad se utiliza si una de las dimensiones del cubo indicadas más arriba está vacía. Una dimensión está vacía si no existe ningún registro de datos de preparación con la combinación del valor del campo de entrada y del valor de destino.

3.5.2 Maquina de Soporte Vectorial (SVM)

Las máquinas de soporte vectorial son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T.
Estos métodos están propiamente relacionados con problemas de clasificación y regresión. Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación definido como el vector entre los 2 puntos, de las 2 clases, más cercanos al que se llama vector soporte. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase. Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta[6].

3.5.3 Random Forest

Random Forest es un algoritmo predictivo que usa la técnica de Bagging para combinar diferentes arboles, donde cada árbol es construido con observaciones y variables aleatorias.

En forma resumida sigue este proceso:

1. Selecciona individuos al azar (usando muestreo con reemplazo) para crear diferentes set de datos.
2. Crea un árbol de decisión con cada set de datos, obteniendo diferentes arboles, ya que cada set contiene diferentes individuos y diferentes variables en cada nodo.
3. Al crear los arboles se eligen variables al azar en cada nodo del arbol, dejando crecer el arbol en profundidad (es decir, sin podar).
4. Predice los nuevos datos usando el "voto mayoritario", donde clasificará como "positivo" si la mayoría de los arboles predicen la observación como positiva.[7]

3.6 CV y ROC Curves

3.6.1 Cross Validation (Validacion Cruzada)

La validación cruzada es un procedimiento más sofisticado que el anterior. En lugar de solo obtener una simple estimación de la efectividad de la generalización; la idea es realizar un análisis estadístico para obtener otras medidas del rendimiento estimado, como la media y la varianza, y así poder entender cómo se espera que el rendimiento varíe a través de los distintos conjuntos de datos. Esta variación es fundamental para la evaluación de la confianza en la estimación del rendimiento. La validación cruzada también hace un mejor uso de un conjunto de datos limitado; ya que a diferencia de la simple división de los datos en uno el entrenamiento y otro de evaluación; la validación cruzada calcula sus estimaciones sobre todo el conjunto de datos mediante la realización de múltiples divisiones e intercambios sistemáticos entre datos de entrenamiento y datos de evaluación.[8]

3.6.2 ROC Curves

Las curvas ROC suelen tener una tasa positiva real en el eje Y y una tasa de falsos positivos en el eje X. Esto significa que la esquina superior izquierda de la gráfica es el punto "ideal", una tasa de falsos positivos de cero y una tasa positiva verdadera de uno. Esto no es muy realista, pero sí significa que un área más grande debajo de la curva (AUC) suele ser mejor. La "pendiente" de las curvas ROC también es importante, ya que es ideal para maximizar la tasa positiva verdadera mientras se minimiza la tasa de falsos positivos.
Las curvas ROC se usan generalmente en la clasificación binaria para estudiar la salida de un clasificador. Para extender la curva ROC y el área ROC a una clasificación multiclase o multi- etiqueta, es necesario binarizar la salida. Se puede dibujar una curva ROC por etiqueta, pero también se puede dibujar una curva ROC considerando cada elemento de la matriz indicadora de etiqueta como una predicción binaria (micro promediación).[9]

4 Descripcion de las Actividades

4.1 Metodologia

4.1.1 Etapa1. Importar el Dataset

En esta face se manejo el conjunto de datos seleccionado para almacenarlos en una variable. El dataset originalmente tiene tres clases, pero durante el procedimiento de almacenamiento solo se consideraron dos de las clases convirtiendo la tercena como parte de las primeras dos. Para ello se fue necesario hacer un conjunto de iteraciones al momento de ir guardando los datos y cuando encontraba una fila con la clase 3, esta tomaba un valor aleatorio entre las dos primeras clases.

```
(132L, 5L)
Clases:
[0 1 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1
 1 1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 0 0 0
 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1 0]
```

Fig1. Clases del dataset (0 = negativo, 1 = positivo)

4.1.2 Etapa2. Pre-processing

- En esta etapa se utilizan los metodos de preprocesamiento de la libreria de Scikit-Learn en donde implementamos el PCA al dataset con los datos originales.

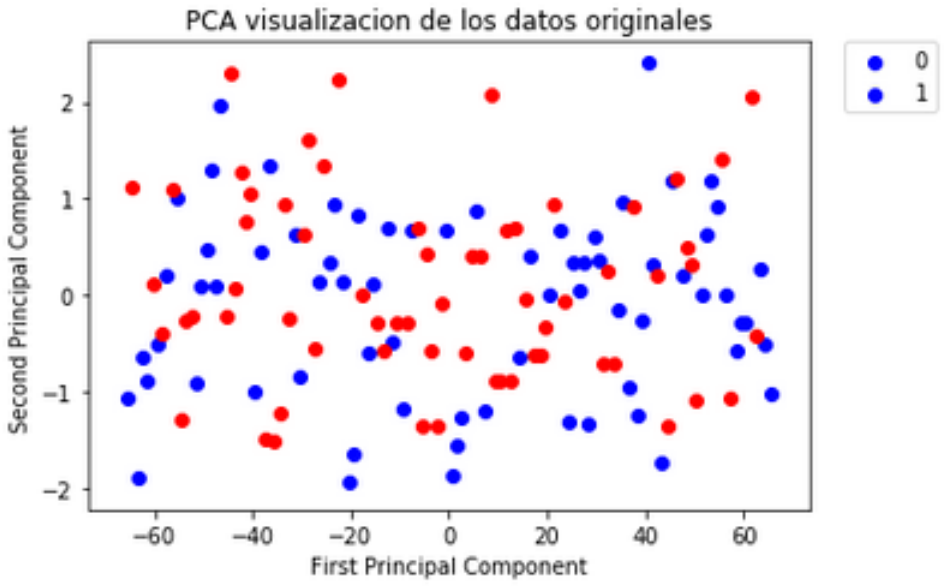

Fig2. Aplicando PCA (0 = azul, 1 = rojo)

- Aqui se muestran la generacion de los datos aplicando el metodo de procesamiento escalado.

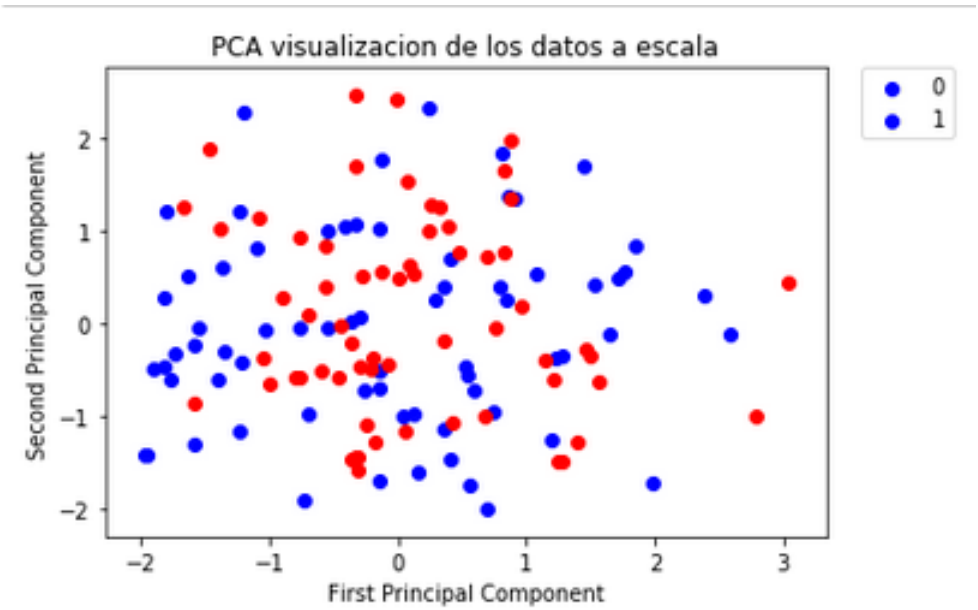

Fig3. Aplicando Escalamiento (0 = azul, 1 = rojo)

- Aqui se muestran la generacion de los datos aplicando el metodo de procesamiento normalizado.

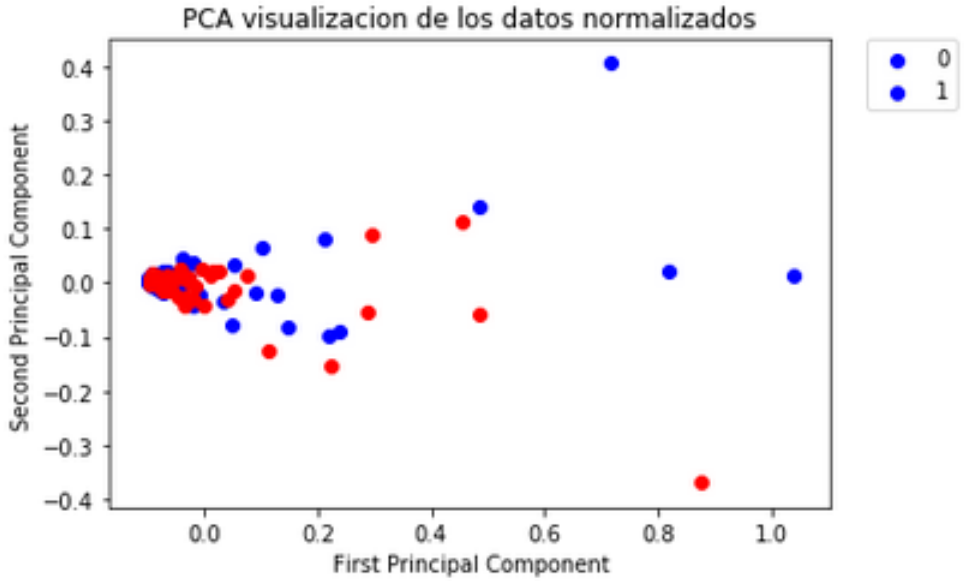

Fig4. Aplicando Escalamiento (0 = azul, 1 = rojo)

- Aqui se muestran la generacion de los datos aplicando el metodo de procesamiento escalado y normalizado.

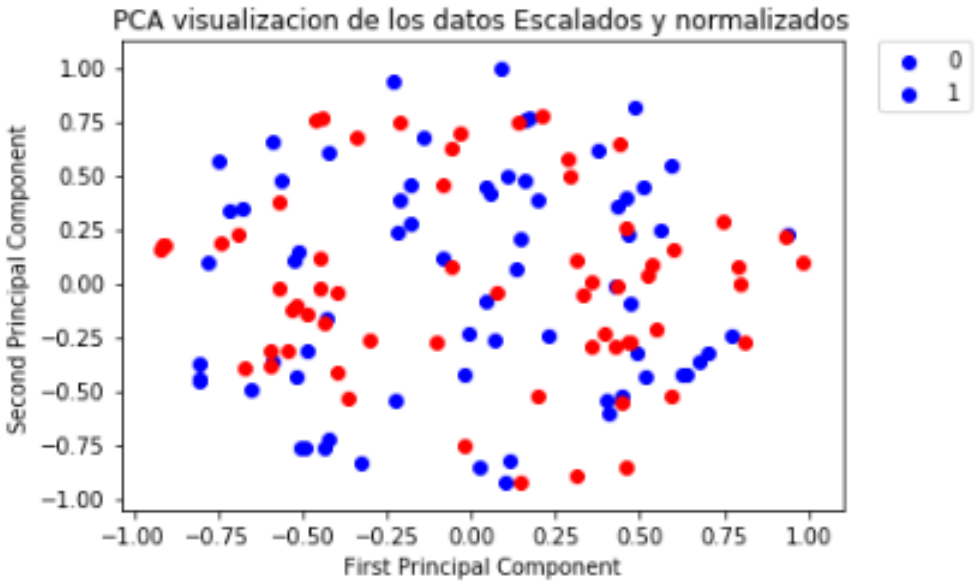

Fig4. Aplicando Escalamiento (0 = azul, 1 = rojo)

4.1.3 Etapa3. Implementacion de Naive Bayes

En esta etapa re trabaja con el algoritmo de clasificacion de Naive Bayes el cual es implementada genericamente en una clase que contiene las funciones correspondientes para el procesamiento de entrada y salida de un conjunto de datos de entrenamiento.
Tambien son utilizados los algoritmos SVM (Maquina de Soporte Vectorial) y RandomForest para la comparacion de resultados junto con el Naive Bayes.

4.2 Analisis de Codigo

```
class Naive_Bayes:

    def __init__(self):
        print "Naive Bayes"

    def fit(self, X, y):
        self.TRAINING_SET = np.array(X) # datos de entrenamiento
        self.SET_CLASES = y # conjunto de clases de entrenamiento
        self.CLASE_POSITIVA = 1 #conjunto de clases sin repetir

        self.FREC_CLASE_POSITIVA = 0 #cantidad de repeticiones por cada clase

        c = 0
        for j in self.SET_CLASES:
            if self.CLASE_POSITIVA == j: self.FREC_CLASE_POSITIVA += 1

        self.PROB_CLASE_POSITIVA = self.FREC_CLASE_POSITIVA / float(len(self.SET_CLASES))

        return self

    def predict_proba(self, X_test):

        PROB_CLASES = []

        #print self.FREC_CLASE_POSITIVA, ',', self.PROB_CLASE_POSITIVA
        #print len(X_test)
        v = 0
        #print self.PROB_CLASE_POSITIVA
        for test in X_test: #INSTANCIA DE PRUEBA
            proba = self.PROB_CLASE_POSITIVA
            #print proba,' ',v

            for j in range(len(self.TRAINING_SET[0])): #recorre todas las filas del conjunto de entrenamiento
                #print 'test',j,': ',test[j]
                cantidad = 0
                for n in range(len(self.SET_CLASES)): #recorre las columnas
                    if self.TRAINING_SET[n][j] == test[j] and self.SET_CLASES[n] == 1:
                        cantidad += 1

                        #print 'clase: '
                #print cantidad
                if cantidad > 0:
                    proba = proba * (cantidad / float(self.FREC_CLASE_POSITIVA))
                #print proba

            PROB_CLASES.append(proba)
            v += 1

        return PROB_CLASES
```

Fig5. Clase de Naive Bayes

En este codigo se muestra la clase de naive bayes la cual contiene dor funciones, la funcion de fit() que es la que obtiene el conjunto de datos de entrenamiento (X) y las clases (y). Esta funcion lo que hace es hacer la seleccion de la clase positiva y calcular la frecuancia con la que se repite. Tambien calcula la probabilidad con la que frecuentemente se repite en las filas del dataset.
Tambien esta la funcion de predict_proba() la cual obtiene un conjunto de datos de prueba los cuales serviran para calcular la probabilidad de que la clase positiva pertenezca a cada uno de los datos.

```
#Algoritmo Naive Bayes
classifier = Naive_Bayes()
tprs = []
aucs = []
mean_fpr = np.linspace(0, 1, 100)

plt.figure(figsize=(15, 10))

i = 0
for train, test in kf.split(X):
    #X_train = pd.DataFrame(np.array(X[train]).reshape(len(X[train]),5), columns = list([0,1,2,3,4]))
    #classifier.fit(X_train, y[train])
    #print 'Entrenamiento: \n',X[train], y[train],'\nPrueba: \n', X[test]
    classifier.fit(X[train], y[train])
    probas_ = classifier.predict_proba(X[test])
    #print 'Probabilidad de Clase:\n',probas_
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_)
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Luck', alpha=.8)

mean_tpr = np.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = np.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = np.std(tprs, axis=0)
tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Naive Bayes')
plt.legend(loc="lower right")
plt.show()
```

Fig6. Uso de K-Fold y Curvas ROC

En este proceso de codigo se utiliza el metodo de K-Fold para obtener el numero de validaciones que se harán para el dataset con sus respectivos datos de prueba, dentro de cada iteracion es donde se hace llamar a las funciones de cada uno de los tres algoritmos de clasificacion con los que se trabaja, tambien utilizando los datos obtenidos de las funciones para la visualizacion de las graficas que muestran el promedio de las curvas ROC y AUC por cada iteracion.

5 Analisis de Resultados

Para definir los datos de entrenamiento del dataset, este fue dividido en 12 rebanadas de las cuales cada una contienen 11 de las filas del dataset y asi, para cada una de las rebanadas se calculara la probabilidad de que se encuentre la clase positiva (clase 1) y esto despues sera reflejado en las graficas que seran generadas con la vista de Curvas de ROC.
En las siguientes graficas se muestran el resultados del promedio de AUC y ROC para cada iteracion procesando el conjunto de datos de entrenamiento por los algoritmos de SVM, Naive Bayes y Random Forest.

5.1 Naive Bayes

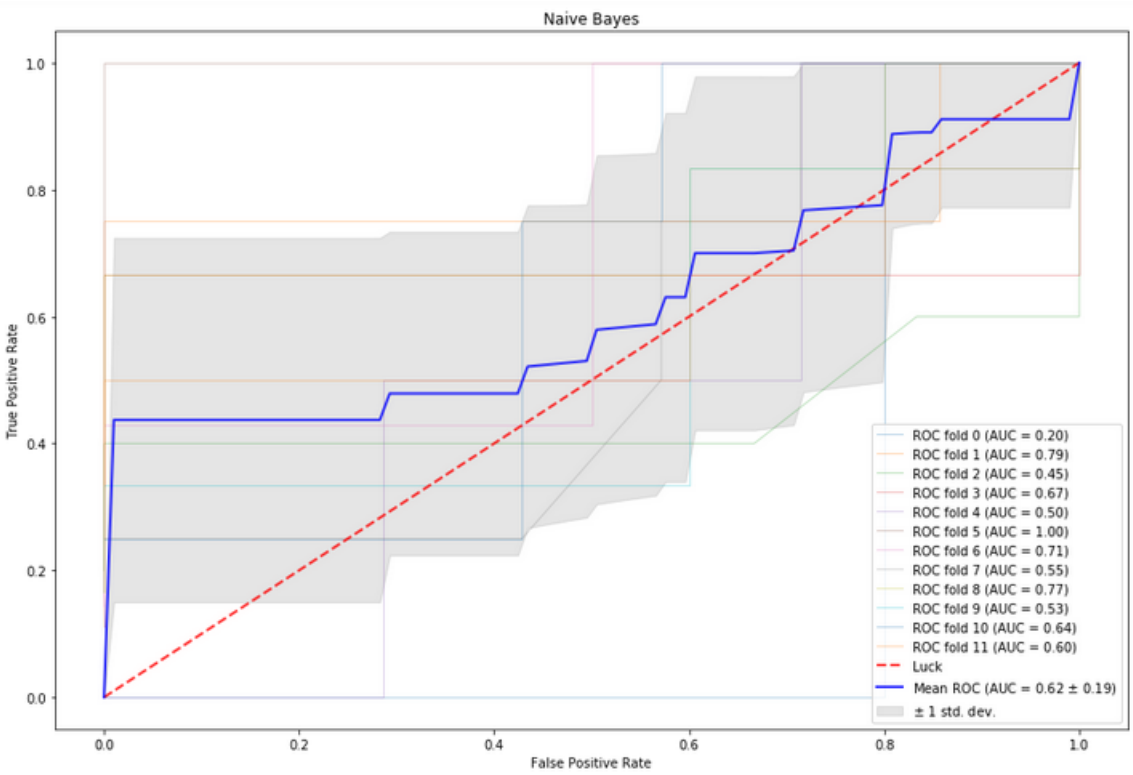

Fig7. Resultados con Naive Bayes

Como se muestra, nuestro algoritmo de Naive Bayes obtuvo estos resultados que al parecer no son tan exactos como se esperaba, pero esto puede que tenga que ver con la estructura del dataset, ya que en una de las columnas que lo contemplan no existen datos repetidos y esto puede que afecte el proceso de clasificacion de los datos al no hallar las probabilidades mas altas de que una clase pertenezca a un conjunto de datos al momento de aplicar el algoritmo de clasificacion.

5.2 Maquina de Soporte Vectorial

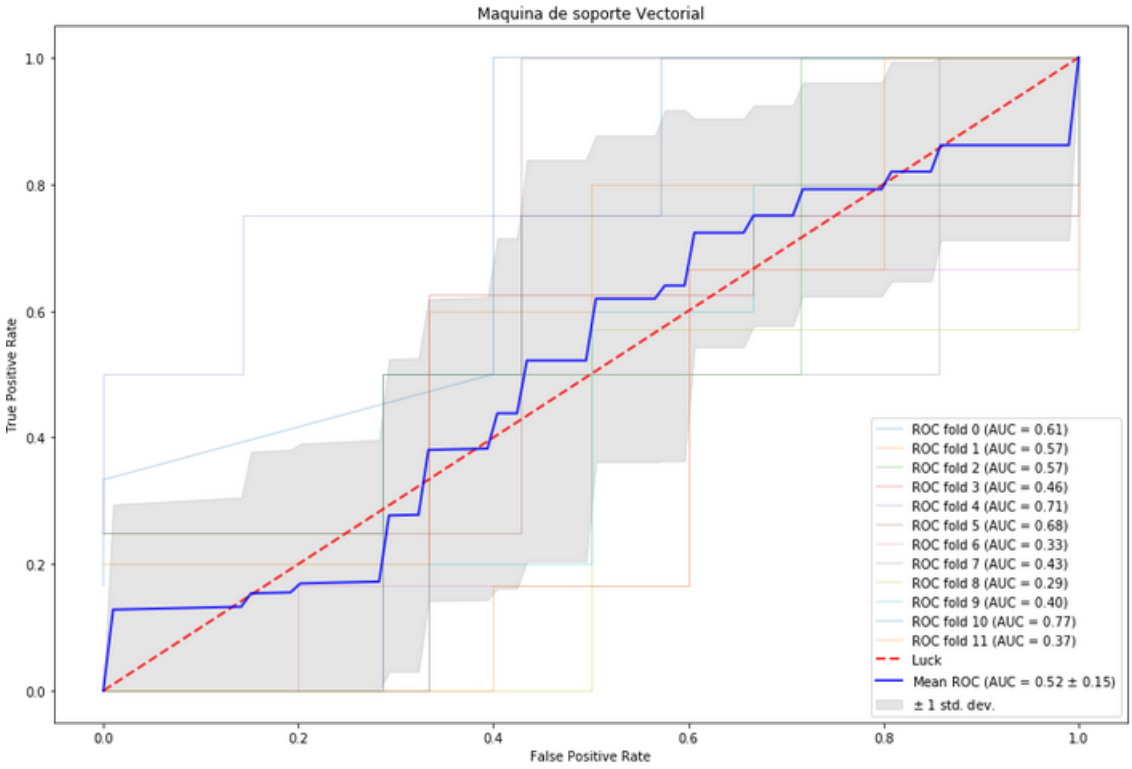

Fig8. Resultados con SVM
Con el resultado del SVM se puede ver que el resultado es aun mas bajo que nuestro algoritmo de Naive Bayes ya que se encuantra muy pegado a la linea con un valor de AUC de 0.52 y una desviacion estandar de 0.15.

5.3 Random Forest

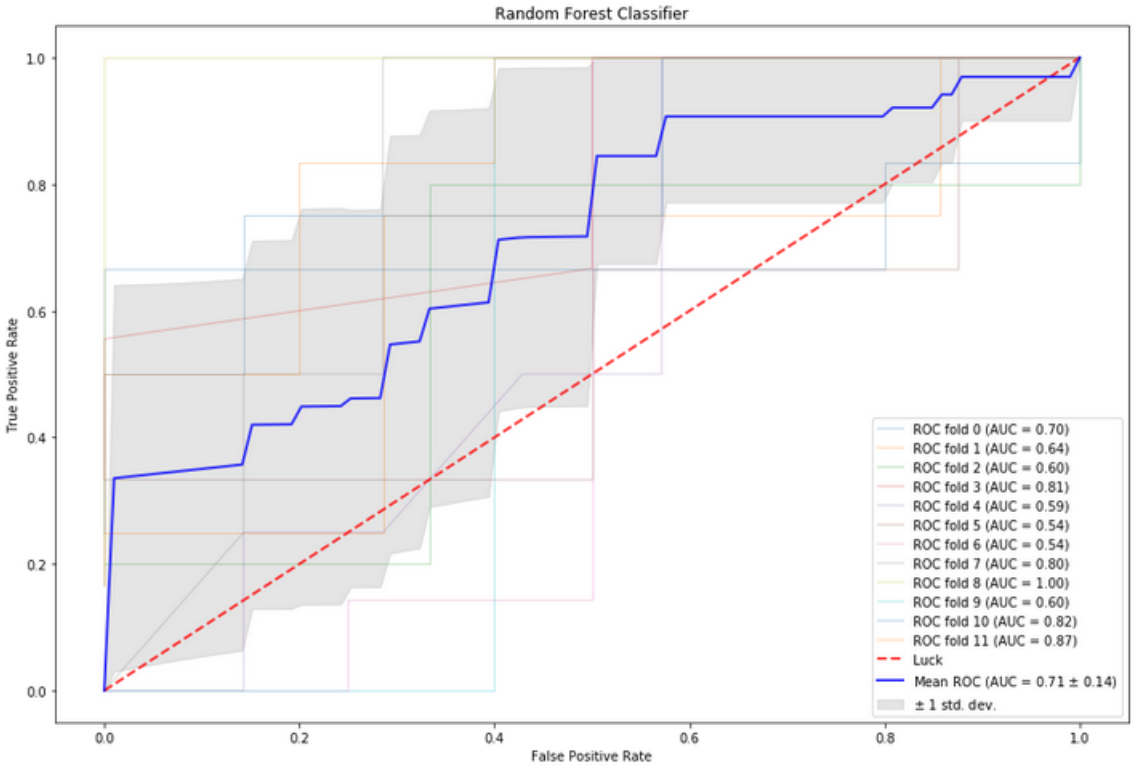

Fig9. Resultados con RandomForest

Por ultimo se muestra el resultado aplicando el Random Forest, el cual segun las probabilidades es el que obtiene mejores resultados de AUC = 0.71 y como desviacion estandar de 0.14 a comparacion de los otros dos.

6 Conclusiones

Finalmente se puede decir que acorde a los resultados obtenidos al aplicar cada uno de estos algoritmos, el algoritmo clasificador que mejor se acopla a este conjunto de datos seleccionado es el de Random Forest ya que obtuvo las mejores probabilidades para AUC y ROC y con la desviacion estandar mas pegada a cero queriendo decir con esto que los resultados de la clasificacion de los datos son mas perfeccionados a comparacion de los de Naive Bayes y SVM.

7 Referencias

[1] https://desarrolloweb.com/articulos/1325.php

[2] http://jupyter.org/

[3] https://jarroba.com/pandas-python-ejemplos-parte-i-introduccion/

[4] https://en.wikipedia.org/wiki/Scikit-learn

[5] https://www.ibm.com/support/knowledgecenter/es/SSEPGG_9.7.0/com.ibm.im.overview.doc/c_nai

[6] https://es.wikipedia.org/wiki/M%C3%A1quinas_de_vectores_de_soporte

[7] http://apuntes-r.blogspot.mx/2014/11/ejemplo-de-random-forest.html

[8] https://relopezbriega.github.io/blog/2015/10/10/machine-learning-con-python/

[9] http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html