Entrega Final de Proyecto

POR:
Jhon Vásquez
Juan Felipe Santa

MATERIA:
Introducción a la Inteligencia Artificial

PROFESOR:
Raul Ramos Pollán

UNIVERSIDAD®
DE ANTIOQUIA

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN

2022

1. Introducción

En el panorama actual de la seguridad cibernética, el aumento exponencial de las amenazas de malware ha planteado desafíos significativos para proteger los sistemas informáticos. Las organizaciones y los usuarios individuales se enfrentan a la constante tarea de identificar y mitigar las diversas formas de software malicioso que buscan infiltrarse en sus redes y dispositivos.

En este contexto, el desarrollo de modelos de Machine Learning se ha convertido en una herramienta fundamental para detectar y prevenir el malware. En este proyecto, nos centramos en el conjunto de datos "Microsoft Malware Prediction", una valiosa recopilación de información proporcionada por Microsoft que aborda precisamente esta problemática.

El conjunto de datos "Microsoft Malware Prediction" contiene una amplia gama de características extraídas de diferentes fuentes, como archivos binarios y registros de eventos, recopilados a lo largo de un período de tiempo significativo. Estas características incluyen información sobre atributos de archivos, configuraciones de hardware y software, comportamientos del sistema operativo, entre otros.

El objetivo principal de este proyecto es utilizar técnicas de Machine Learning para construir un modelo capaz de predecir la presencia de malware en un sistema informático. Al explorar y analizar exhaustivamente el conjunto de datos, podremos extraer patrones y relaciones ocultas que nos permitirán construir un modelo de clasificación preciso y confiable.

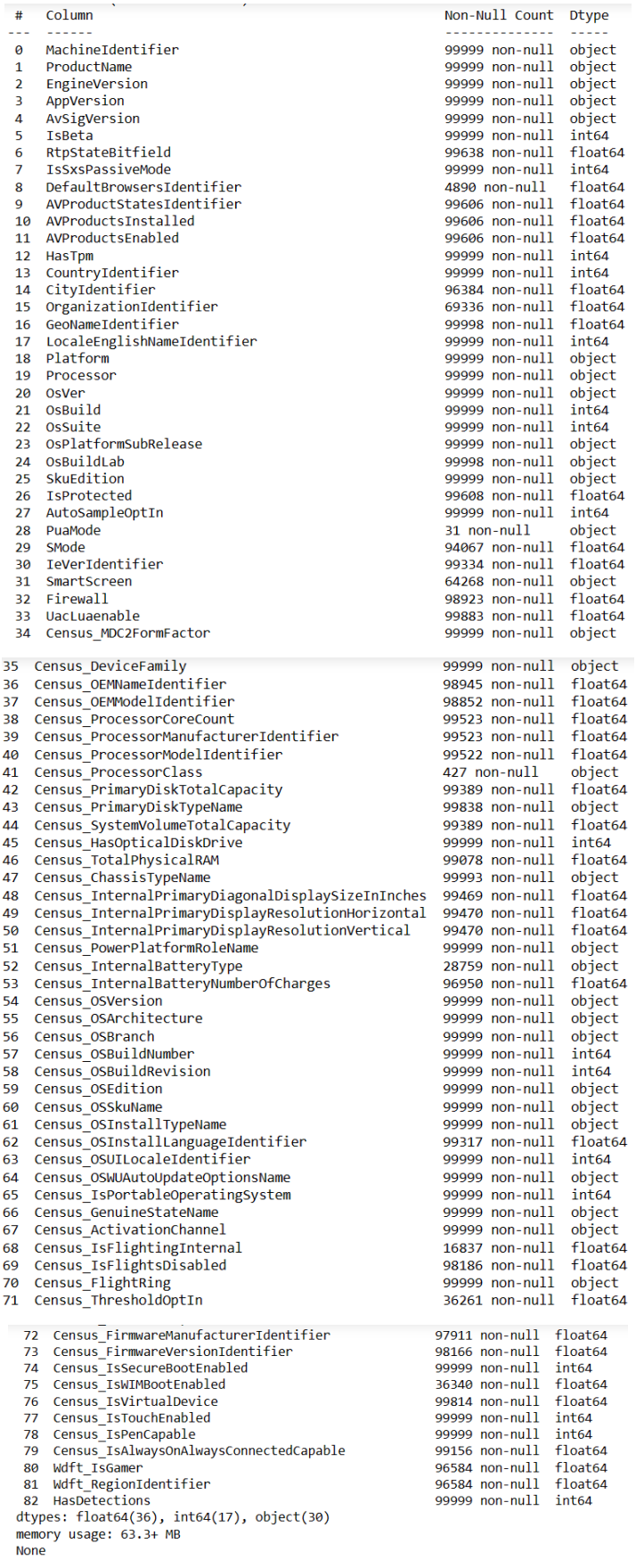
La importancia de este proyecto radica en su capacidad para fortalecer la seguridad cibernética, ya sea a nivel empresarial o individual. Al tener un modelo de detección de malware preciso, las organizaciones y los usuarios podrán anticipar y mitigar las amenazas potenciales, protegiendo sus sistemas y datos sensibles de posibles ataques.

En este documento se presenta el trabajo realizado donde se hace un preprocesado de datos y se implementa dos Modelos supervisados (Decision Tree y Random Forest Classifier). Además, se hace combinación del algoritmo no supervisado llamado PCA con los algoritmos supervisados mencionados anteriormente.

2. Exploración descriptiva del dataset

El dataset lo subió Microsoft al sitio de Kaggle para una competición llamada “Microsoft Malware Prediction”. El dataset fue recopilado usando la herramienta Windows Defender, la cuál está activa en la mayoría de computadores con el sistema operativo Windows. Cada fila de datos corresponde a una máquina, identificada por la variable MachineIdentifier. La variable que dice si un computador ha sido infectado o no por un malware se llama HasDetections. Como el dataset cuenta con más de ocho millones de filas, se hace una submuestra de los datos, tomando solo cien mil filas.

Las columnas del dataset están descritas de está manera:



```
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   MachineIdentifier                             99999 non-null  object
 1   ProductName                                   99999 non-null  object
 2   EngineVersion                                 99999 non-null  object
 3   AppVersion                                    99999 non-null  object
 4   AvSigVersion                                  99999 non-null  object
 5   IsBeta                                        99999 non-null  int64
 6   RtpStateBitfield                              99638 non-null  float64
 7   IsSxsPassiveMode                              99999 non-null  int64
 8   DefaultBrowsersIdentifier                     4890 non-null   float64
 9   AVProductStatesIdentifier                     99606 non-null  float64
 10  AVProductsInstalled                           99606 non-null  float64
 11  AVProductsEnabled                             99606 non-null  float64
 12  HasTpm                                        99999 non-null  int64
 13  CountryIdentifier                             99999 non-null  int64
 14  CityIdentifier                                96384 non-null  float64
 15  OrganizationIdentifier                        69336 non-null  float64
 16  GeoNameIdentifier                             99998 non-null  float64
 17  LocaleEnglishNameIdentifier                   99999 non-null  int64
 18  Platform                                      99999 non-null  object
 19  Processor                                     99999 non-null  object
 20  OsVer                                         99999 non-null  object
 21  OsBuild                                       99999 non-null  int64
 22  OsSuite                                       99999 non-null  int64
 23  OsPlatformSubRelease                          99999 non-null  object
 24  OsBuildLab                                    99998 non-null  object
 25  SkuEdition                                    99999 non-null  object
 26  IsProtected                                   99608 non-null  float64
 27  AutoSampleOptIn                               99999 non-null  int64
 28  PuaMode                                       31 non-null     object
 29  SMode                                         94067 non-null  float64
 30  IeVerIdentifier                               99334 non-null  float64
 31  SmartScreen                                   64268 non-null  object
 32  Firewall                                      98923 non-null  float64
 33  UacLuaenable                                  99883 non-null  float64
 34  Census_MDC2FormFactor                         99999 non-null  object
 35  Census_DeviceFamily                           99999 non-null  object
 36  Census_OEMNameIdentifier                      98945 non-null  float64
 37  Census_OEMModelIdentifier                     98852 non-null  float64
 38  Census_ProcessorCoreCount                     99523 non-null  float64
 39  Census_ProcessorManufacturerIdentifier        99523 non-null  float64
 40  Census_ProcessorModelIdentifier               99522 non-null  float64
 41  Census_ProcessorClass                         427 non-null    object
 42  Census_PrimaryDiskTotalCapacity               99389 non-null  float64
 43  Census_PrimaryDiskTypeName                    99838 non-null  object
 44  Census_SystemVolumeTotalCapacity              99389 non-null  float64
 45  Census_HasOpticalDiskDrive                    99999 non-null  int64
 46  Census_TotalPhysicalRAM                       99078 non-null  float64
 47  Census_ChassisTypeName                        99993 non-null  object
 48  Census_InternalPrimaryDiagonalDisplaySizeInInches  99469 non-null  float64
 49  Census_InternalPrimaryDisplayResolutionHorizontal  99470 non-null  float64
 50  Census_InternalPrimaryDisplayResolutionVertical    99470 non-null  float64
 51  Census_PowerPlatformRoleName                  99999 non-null  object
 52  Census_InternalBatteryType                    28759 non-null  object
 53  Census_InternalBatteryNumberOfCharges         96950 non-null  float64
 54  Census_OSVersion                              99999 non-null  object
 55  Census_OSArchitecture                         99999 non-null  object
 56  Census_OSBranch                               99999 non-null  object
 57  Census_OSBuildNumber                          99999 non-null  int64
 58  Census_OSBuildRevision                        99999 non-null  int64
 59  Census_OSEdition                              99999 non-null  object
 60  Census_OSSkuName                              99999 non-null  object
 61  Census_OSInstallTypeName                      99999 non-null  object
 62  Census_OSInstallLanguageIdentifier            99317 non-null  float64
 63  Census_OSUILocaleIdentifier                   99999 non-null  int64
 64  Census_OSWUAutoUpdateOptionsName              99999 non-null  object
 65  Census_IsPortableOperatingSystem              99999 non-null  int64
 66  Census_GenuineStateName                       99999 non-null  object
 67  Census_ActivationChannel                      99999 non-null  object
 68  Census_IsFlightingInternal                    16837 non-null  float64
 69  Census_IsFlightsDisabled                      98186 non-null  float64
 70  Census_FlightRing                             99999 non-null  object
 71  Census_ThresholdOptIn                         36261 non-null  float64
 72  Census_FirmwareManufacturerIdentifier         97911 non-null  float64
 73  Census_FirmwareVersionIdentifier              98166 non-null  float64
 74  Census_IsSecureBootEnabled                    99999 non-null  int64
 75  Census_IsWIMBootEnabled                       36340 non-null  float64
 76  Census_IsVirtualDevice                        99814 non-null  float64
 77  Census_IsTouchEnabled                         99999 non-null  int64
 78  Census_IsPenCapable                           99999 non-null  int64
 79  Census_IsAlwaysOnAlwaysConnectedCapable       99156 non-null  float64
 80  Wdft_IsGamer                                  96584 non-null  float64
 81  Wdft_RegionIdentifier                         96584 non-null  float64
 82  HasDetections                                 99999 non-null  int64
dtypes: float64(36), int64(17), object(30)
memory usage: 63.3+ MB
None
```

Figura 1. Columnas del dataset

Los datos faltantes se pueden observar en la siguiente imagen:

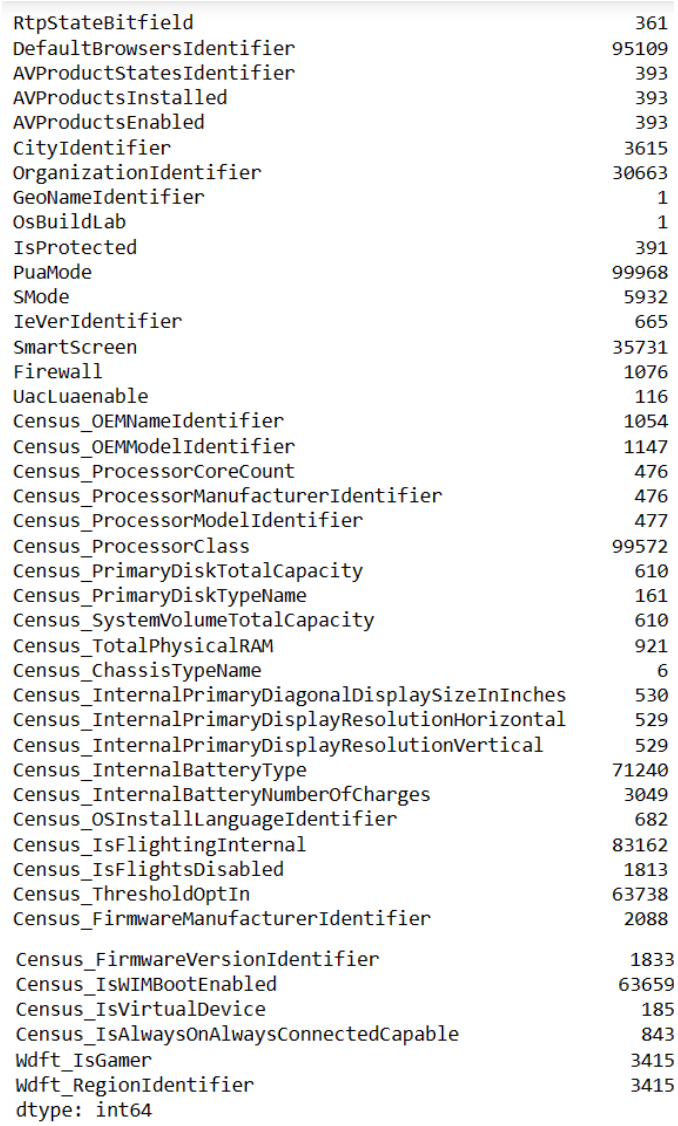


Figura 2. Datos Faltantes

3. Iteraciones de Desarrollo

3.1 Preprocesado de datos

Se realizó una submuestra del conjunto de datos original, lo que resultó en un conjunto de datos con un total de 20,000 filas. Posteriormente, se verificó la presencia de valores faltantes en todas las columnas. Se encontró que la columna *"PuaMode"* tenía la mayor cantidad de valores faltantes, mientras que la columna *"Census_OEMNameIdentifier"* presentaba la menor cantidad de valores faltantes. Además, se verificó los tipos de datos de cada columna, obteniendo presencia de datos de tipo int64, float64 y object (ver figura 1). El porcentaje de valores faltantes se observa en la figura 2, donde el 6% de las columnas del dataset tiene un porcentaje de valores faltantes mayor al 70%; dichas columnas son eliminadas del dataset.

MachineIdentifier	object
ProductName	object
EngineVersion	object
AppVersion	object
AvSigVersion	object
IsBeta	int64
RtpStateBitfield	float64
IsSxsPassiveMode	int64
DefaultBrowsersIdentifier	float64
AVProductStatesIdentifier	float64
AVProductsInstalled	float64
AVProductsEnabled	float64
HasTpm	int64
CountryIdentifier	int64

Figura 3. Algunos columnas del dataset y sus tipos de datos

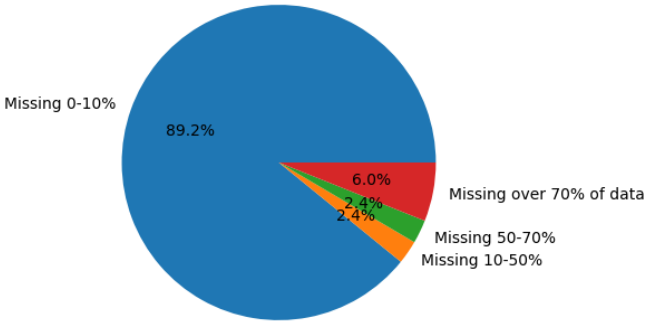


Figura 4. Porcentaje de columnas con un valor de datos faltantes

Se realizó la matriz de correlación, donde se observó que la mayoría de variables tienen correlación positiva débil (ver figura 3). Además, en la correlación respecto a la variable objetivo, se observa que hay variables que dan resultado NaN (Ver figura 4).

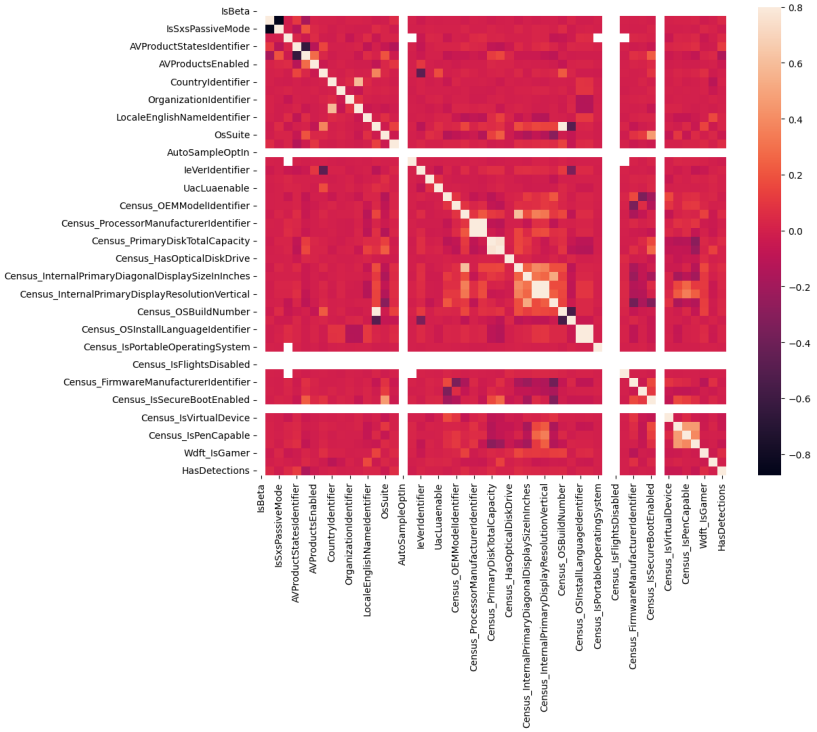


Figura 5. Matriz de Correlación

AVProductsInstalled	-0.145669
IsBeta	NaN
AutoSampleOptIn	NaN
Census_IsFlightingInternal	NaN
Census_IsFlightsDisabled	NaN
Census_IsWIMBootEnabled	NaN

Figura 6. Valores NaN en correlación

Para las variables categóricas se llenan los datos faltantes con la moda y se aplica la técnica de One-Hot Encoding, lo cuál hace que la dimensión del dataset aumente considerablemente, teniendo ahora 2953 columnas.

Por otra parte, se analiza la distribución de las clases, encontrando que el dataset se encuentra muy balanceado, teniendo 10019 de máquinas que han sido infectadas y 9980 que no (ver figura 5)

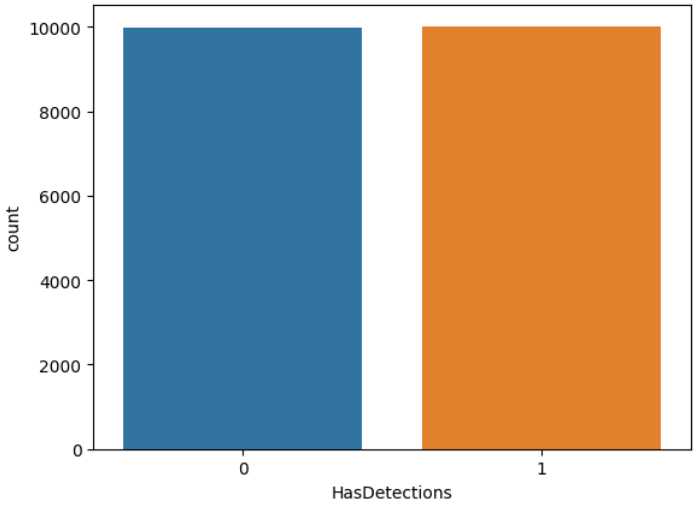


Figura 7. Valores NaN en correlación

3.2 Modelos Supervisados

Se seleccionaron los algoritmos: Decision Tree y Random Forest para realizar el respectivo entrenamiento de los datos y consecuentemente generar modelos predictivos. Para todos los algoritmos se usa la librería Sklearn, la cuál ya tiene varios de ellos implementados. Con la finalidad de encontrar los mejores hiperparámetros para cada algoritmo, se usó el método *GridSearchCV.*

3.2.1 Decision Tree

Para este algoritmo se dan valores para el hiperparámetro "mas_depth" de 3,5,10,15,20 y 50. En el siguiente código se puede ver la implementación para la selección de los mejores hiperparametros:

```
parametros = {'max_depth': [3,5,10,15,20,50]}

estimatorDT = DecisionTreeClassifier()


decission_tree = GridSearchCV(estimator = estimatorDT,
                        param_grid = parametros,
                        cv = ShuffleSplit(n_splits= 10, test_size=val_size),
                        scoring = 'roc_auc',
                        verbose = 1,
                        return_train_score = True,
                        n_jobs = -1)

decission_tree.fit(Xtv, ytv)
```

Figura 8. Búsqueda hiperparámetros (Decision Tree)

3.2.2 Random Forest Classifier

Dado que el problema al que se enfrenta es de clasificación, se usa el algoritmo de Random Forest Classifier. Para buscar los mejores hiperparámetros se dan los siguientes parámetros al método *GridSearchCV: 'n_estimators': [5,10,20,50]* y *'max_depth':[2,5,8,12,15]*. A continuación se puede observar la implementación:

```
parametros = { 'n_estimators': [5,10,20,50],
               'max_depth':[2,5,8,12,15]}

estimatorRF = RandomForestClassifier()

forest_reg = GridSearchCV(estimator = estimatorRF,
                          param_grid = parametros,
                          cv = ShuffleSplit(n_splits= 10, test_size=val_size),
                          scoring = 'roc_auc',
                          verbose = 1,
                          return_train_score = True,
                          n_jobs = -1)
forest_reg.fit(Xtv, ytv)
```

Figura 9. Búsqueda hiperparámetros (Random Forest Classifier)

3.3 Modelos No Supervisados

Se opta por usar PCA para reducir la dimensionalidad. Para ello, se realizan pruebas con 10, 50, 100, 500, 1000 y 1500 componentes. Para la implementación se hace una combinación de un algoritmo supervisado + PCA. A continuación se puede observar las implementaciones de un Decision Tree + PCA y un Random Forest + PCA.

PCA + Decision Tree

```
[ ]  components = [10, 50, 100, 500, 1000, 1500]
     perf = [] #desempeños de los modelos
     DTC = DecisionTreeClassifier(max_depth=5)
     for i in components:
         pca = PCA(n_components = i)
         X_t = pca.fit_transform(Xtv)

         #Partición de datos
         XtvPCA, XtsPCA, ytvPCA, ytsPCA = train_test_split(X_t, ytv, test_size=val_size)
         print (XtvPCA.shape, XtsPCA.shape)

         DTC.fit(XtvPCA, ytvPCA)

         y_pred = DTC.predict(XtsPCA)

         # Calcular el rendimiento utilizando ROC AUC
         roc_auc = roc_auc_score(ytsPCA, y_pred)

         perf.append(roc_auc)
         print('ROC_AUC del modelo con ', i ,'elementos: ',"{:.5f}".format(roc_auc))
         print('------------')

     print('Mejor ROC_AUC: ',"{:.5f}".format(np.min(perf)),'  ; obtenido con ',components[np.argmin(perf)],' componentes para PCA')
```

Figura 10. Implementación PCA + Decision Tree

PCA + Random Forest

```
components = [10, 50, 100, 500, 1000, 1500]
perf = [] #desempeños de los modelos
Rdm_forest = RandomForestClassifier(n_estimators = 50,max_depth = 8)
for i in components:
    pca = PCA(n_components = i)
    X_t = pca.fit_transform(Xtv)

    #Partición de datos
    XtvPCA, XtsPCA, ytvPCA, ytsPCA = train_test_split(X_t, ytv, test_size=val_size)
    print (XtvPCA.shape, XtsPCA.shape)

    Rdm_forest.fit(XtvPCA, ytvPCA)

    y_pred = Rdm_forest.predict(XtsPCA)

    # Calcular el rendimiento utilizando ROC AUC
    roc_auc = roc_auc_score(ytsPCA, y_pred)

    perf.append(roc_auc)
    print('ROC_AUC del modelo con ', i ,'elementos: ',"{:.5f}".format(roc_auc))
    print('------------')

print('Mejor ROC_AUC: ',"{:.5f}".format(np.min(perf)),'  ; obtenido con ',components[np.argmin(perf)],' componentes para PCA')
```

Figura 11. Implementación PCA + Random Forest

3.4 Resultados, métricas y curvas de aprendizaje

3.4.1 Métrica

Se elige la métrica ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) para nuestro problema de clasificación. Esta métrica es ampliamente usada en problemas de clasificación binaria dado a que tiene en cuenta tanto la tasa de verdaderos positivos como la tasa de falsos positivos en un clasificador binario. Esto es particularmente útil en escenarios donde es importante equilibrar la sensibilidad a ambos tipos de errores. Las puntuaciones de AUC por debajo de 0,5 se consideran malas, y las puntuaciones entre 0,5 y 0,7 se consideran medias. Puntuaciones más altas que 0.7 son consideradas buenas.

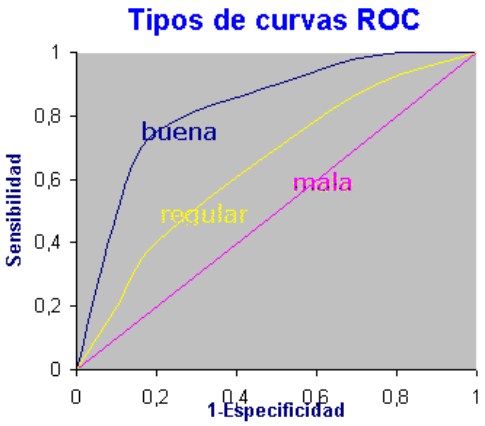


Figura 12. Curva ROC

3.4.2 Mejores hiperparámetros

Luego de la búsqueda de los mejores hiperparámetros con el método *GridSearchCV,* se obtienen los siguientes resultados:

Módelo	Mejores Hiperparámetros
Decision Tree	max_depth=5
Random Forest	max_depth=8, n_estimators=50
Decision Tree + PCA	max_depth=5
Random Forest + PCA	max_depth=8, n_estimators=50

Tabla 1. Mejores hiperparámetros

3.4.3 Rendimiento

Luego de probar los modelos con sus mejores hiperparámetros se obtienen los siguientes resultados de rendimiento.

Módelo	ROC-AUC
Decision Tree	0.5830358258956474
Random Forest	0.5863321583039577
Decision Tree + PCA	0.5741368534213355
Random Forest + PCA	0.5605827645691143

Tabla 2. Resultados de rendimiento con ROC_AUC_SCORE

3.4.4 Curvas de Aprendizaje

Las cuatro curvas de aprendizaje de los modelos seleccionados se pueden observar a continuación:

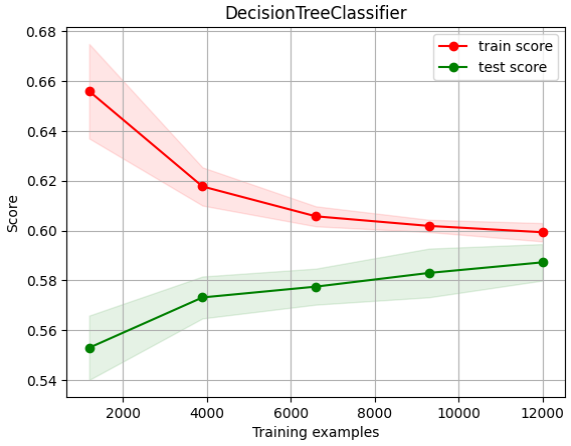


Figura 13. Curva de Aprendizaje (Decision Tree)

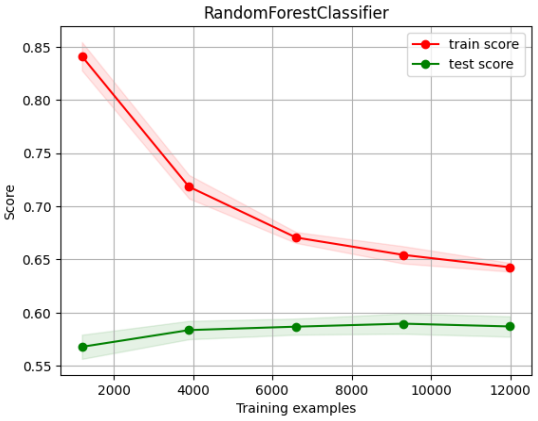


Figura 14. Curva de Aprendizaje (Random Forest)

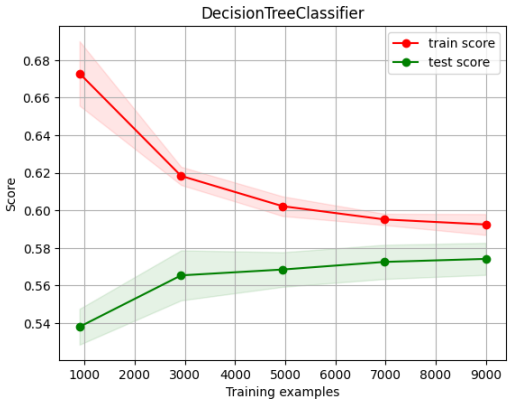


Figura 15. Curva de Aprendizaje (Decision Tree + PCA)

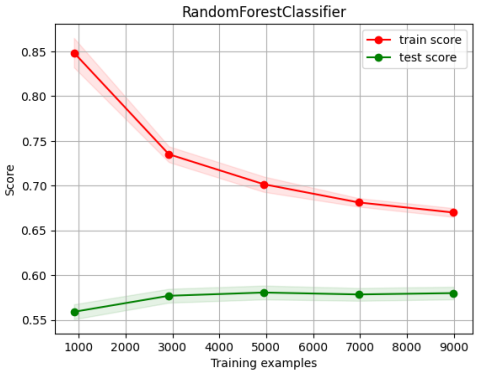


Figura 16. Curva de Aprendizaje (Random Forest + PCA)

Se puede observar en todas las curvas de aprendizaje que existe un gran sesgo, además que el rendimiento fue bastante regular. Una posible solución para esto es agregar más columnas al dataset o volver más complejos los modelos. También se considera que faltó un tratamiento mejor de la información en el procesamiento de la data que pudiera mejorar el rendimiento de los modelos, como lo es eliminar columnas que tengan poca correlación con la variable objetivo.

4. Retos y consideraciones de despliegue

En cuanto a la recopilación de información no hay muchos inconvenientes, ya que Microsoft seguirá recopilando la información de las máquinas constantemente. De igual manera, Microsoft tendrá toda la infraestructura para seguir con la operación del modelo.

5. Conclusiones

Se observa la importancia de un buen preprocesado de datos, ya que esta primera fase es crucial para obtener mejores resultados en el rendimiento de los modelos. También se puede notar que hace falta implementar más modelos para tener más variedad de opciones para elegir un modelo para llevar a producción. También es importante probar con más hiperparametros los modelos, para así realizar una mejor selección de los mismos.