3° Presentación Deep learning

Presentadores

Alan Acuña

Felipe Mena

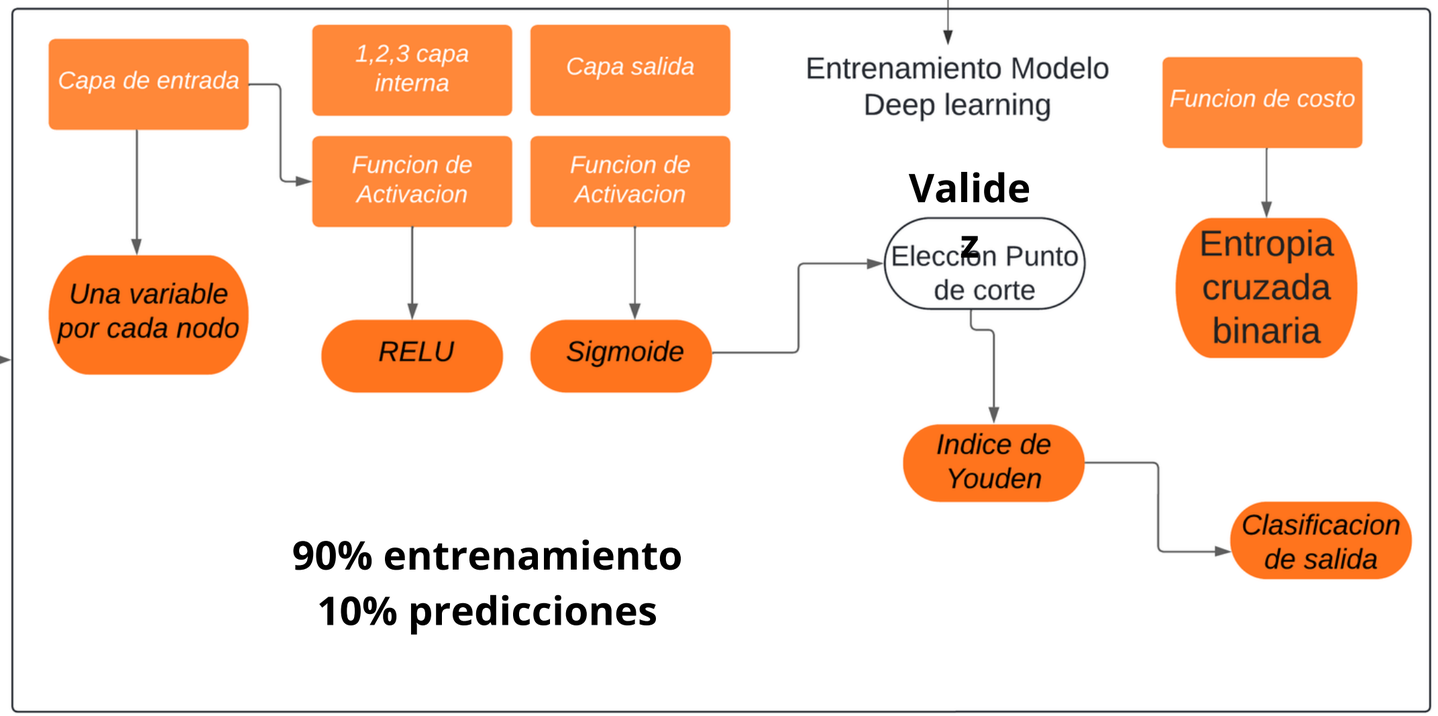
Objetivo:

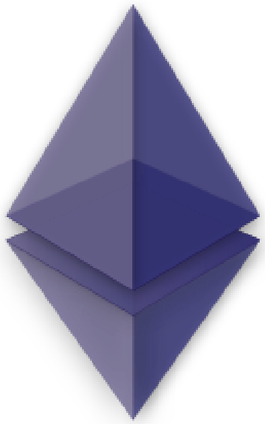
Identificar cuentas fraudulentas en el contexto de criptomonedas de la plataforma ethereum comparando el poder predictivo de modelo de redes neuronales con el modelo SVM y modelo logistico , en base a esto escoger el modelo que mejor clasifica.

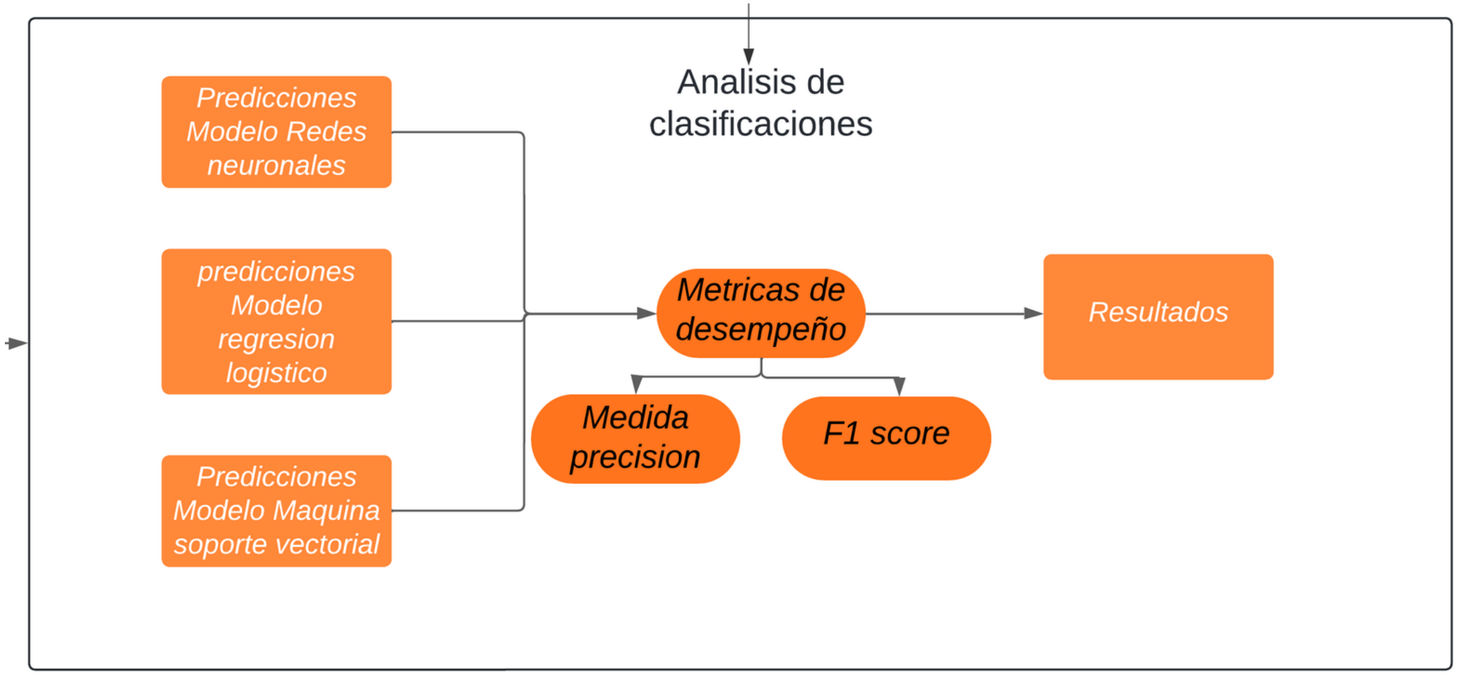
Hipotesis:

Evaluar si el modelo de redes neuronales de tres capas internas tiene un mejor desempeño en predecir frente al modelo logístico y SVM

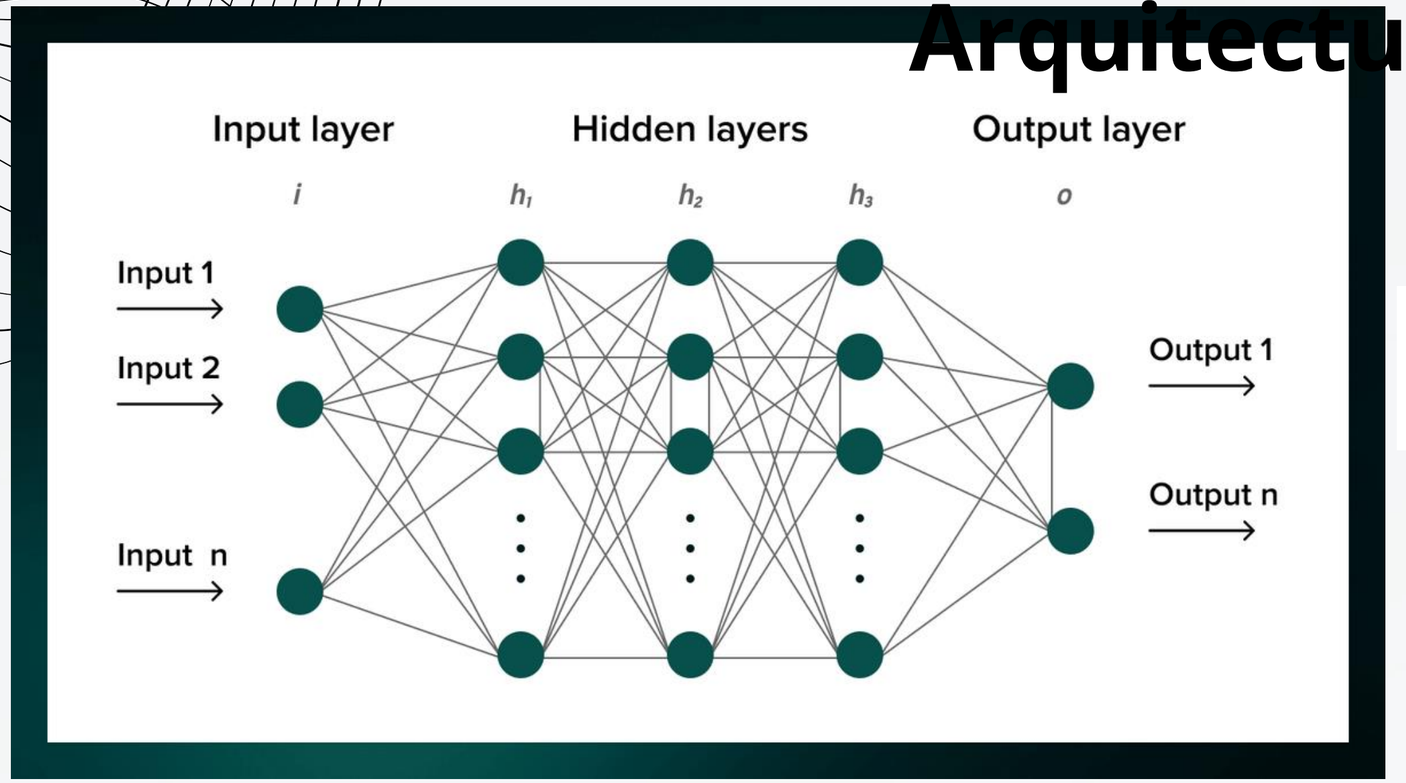
Esquema Metodologico







Definición de Arquitectura

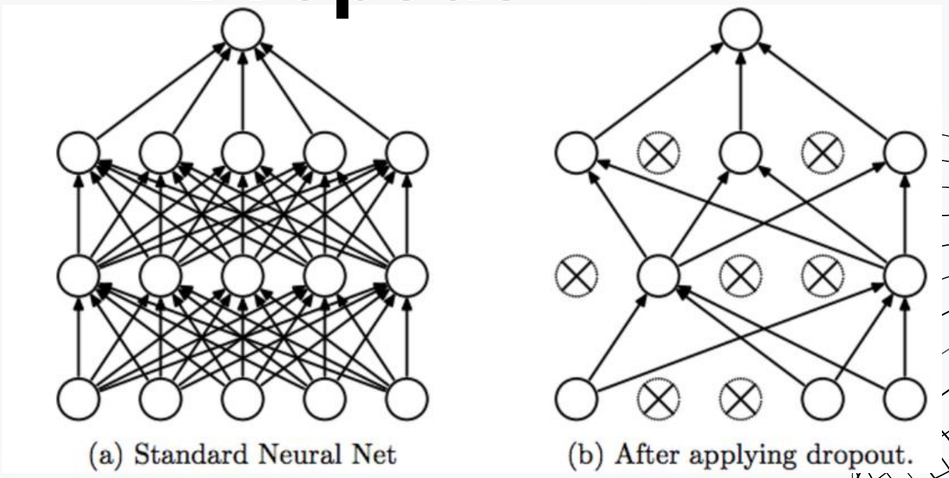


- Capa de entrada: 5 neuronas
- 1 capa interna: 50 neuronas|tasa 30%
- 2 capa interna: 32 neuronas|tasa 30%
- 3 capa interna: 20 neuronas|tasa 30%
- Capa de salida: 1 neurona

Función de activación

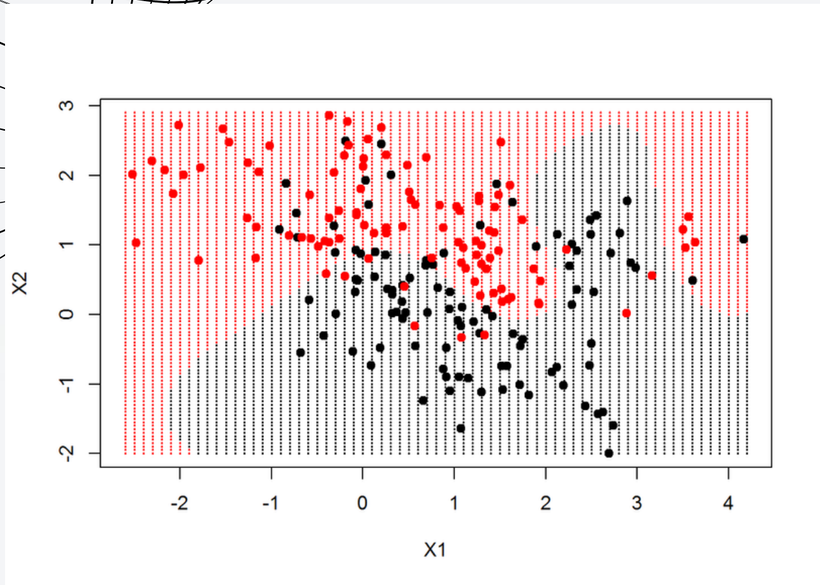
$$f(Z) = \frac{1}{1 + exp(-Z)} \quad \phi(v) = \max\{0, v\}$$

Método Regularizador Dropout



Definición de Arquitectura

Modelo SVM



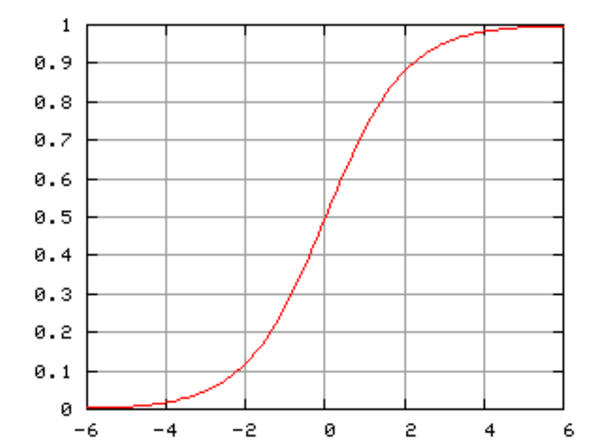
$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

Kernel:Radial

Gamma:10

Función Costo:20

Modelo Regresión logistica

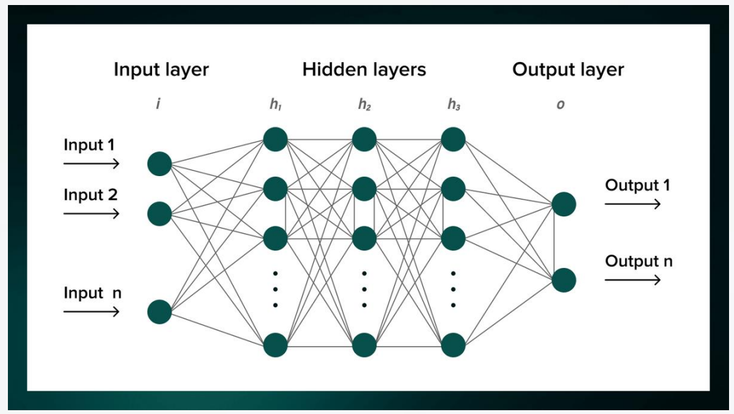


$$\mathbb{P}(Y = 1|X_1, .., X_k) = \frac{e^{\beta_0 + \vec{x}\beta}}{1 + e^{\beta_0 + \vec{x}\beta}} \quad \mathbb{P}(Y = 0|X_1, .., X_k) = \frac{1}{1 + e^{\beta_0 + \vec{x}\beta}}$$

Descripción del proceso de entrenamiento

Modelos de redes neuronales

Función de Costo: $\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$



Épocas: 200

learning rate= 0.001

Algoritmo de optimización: Adam $w_{t+1} = w_t - \alpha_t \frac{\tilde{u}}{\sqrt{\tilde{v}+\varepsilon}}$

beta 1 = 0.9

beta 2 = 0.999

Tiempo de cómputo: 4.09 minutos

Procesador PC: AMD Ryzen 5 2500U

Metricas:

$$Precision = \frac{VP}{VP + FP}$$

$$F1_{score} = 2\frac{Precision * S}{Presicion + S}$$

	Valores Predichos		
Valores Reales	Negativo $\hat{Y} = 0$	Positivo $\hat{Y} = 1$	Totales
Negativo Y=0	Verdaderos Negativo (VN)	Falsos Positivo (FP)	VN+FP
Positivo Y=1	Falso Negativo (FN)	Verdadero Positivo (VP)	FN+VP

Descripción del proceso de entrenamiento

Modelo/ Descripción	Modelo SVM y logístico SVM	Logístico
Tiempo de computo	**20.44 Segundos**	**3.45 Segundos**

SVM

Código en R

```
cost_svm=tune(svm,factor(fraude_train[,1])~fraude_train[,2]+fraude_train[,3]+fraude_train[,4]+
     fraude_train[,5],data=fraude_train,
      kernel = 'radial',ranges=list(cost = c(20),gamma = c(0.20,0.5, 1, 2, 3, 4, 5, 10)))
plot(cost_svm)

tiempo_ini_SVM= Sys.time()
mod_svm=svm(as.factor(fraude_train[,1])~.,data=fraude_train[,-1],kernel='radial'
            ,cost=20,gamma=10,decision.values=T,probability=T)
tiempo_fin_SVM= Sys.time()
tiempo_SVM=tiempo_ini_SVM-tiempo_fin_SVM
```

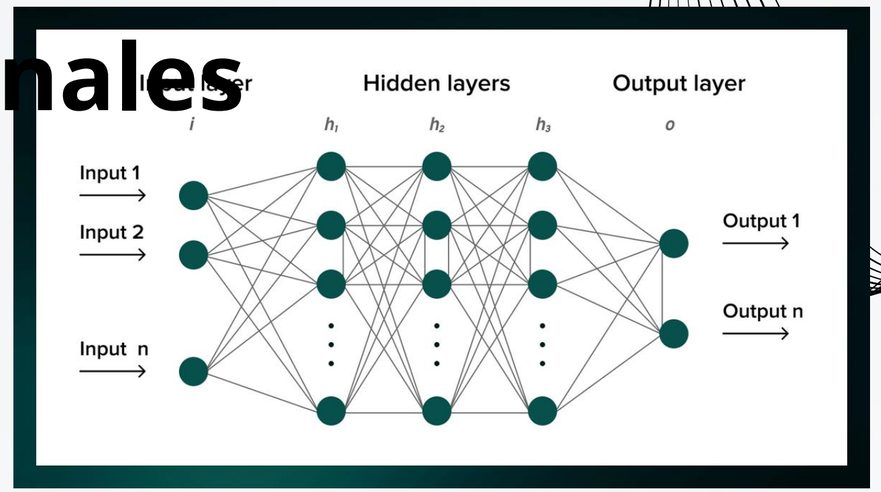
Logistico:

```
tiempo_ini_log= Sys.time()
mod_logistic=glm(fraude_train[,1]~fraude_train[,2]+fraude_train[,3]+fraude_train[,4]+
                   fraude_train[,5]+fraude_train[,6],family=binomial(link='logit'))
tiempo_fin_log= Sys.time()
```

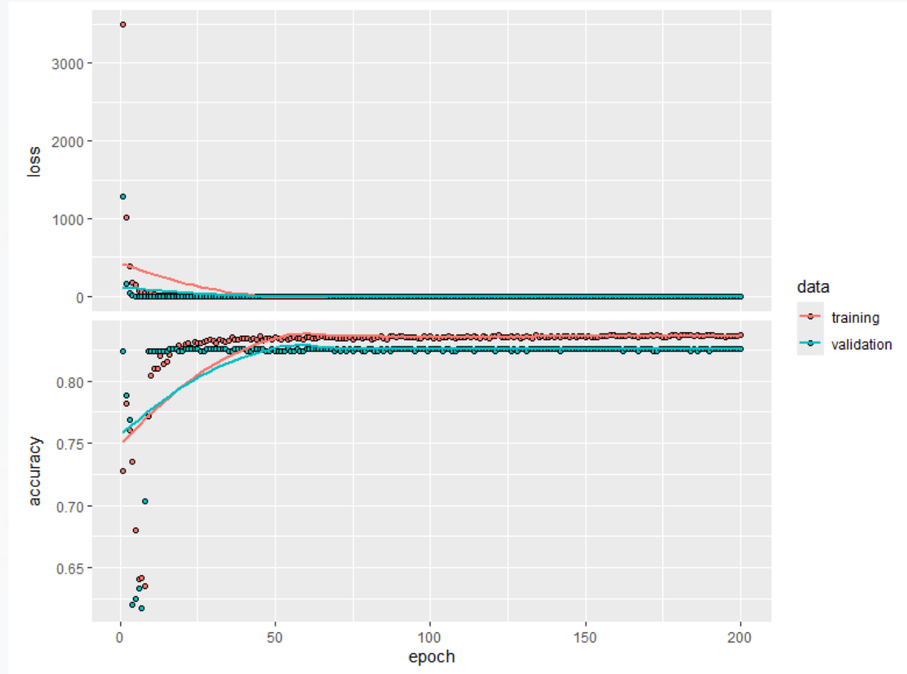
Proceso de entrenamiento y validación

Código de Modelo de Redes Neuronales R:

```
mod_RN2=keras_model_sequential()

mod_RN2%>%layer_dense(units = 5,activation = 'relu',input_shape = c(5))%>%

  layer_dense(units = 50,activation = 'relu')%>%layer_dropout(rate=0.3)%>%
  layer_dense(units=32,activation='relu')%>%
  layer_dropout(rate=0.3)%>%
layer_dense(units=20,activation='relu')%>%
layer_dropout(rate=0.3)%>%
  layer_dense(units=1,activation='sigmoid')
```




```
mod_RN2%>%compile(loss = 'binary_crossentropy',metrics=c('accuracy'),optimizer=optimizer_adam()
    )
tiempo_ini_RNN= Sys.time()

historia=mod_RN2%>%fit(as.matrix(fraude_train[,-1]),as.matrix(fraude_train[,1]),epochs=200,
    batch_size = 100,validation_split=0.10
,callbacks = callback)
plot(historia)

tiempo_fin_RNN= Sys.time()
tiempo_fin_RNN-tiempo_ini_RNN
```

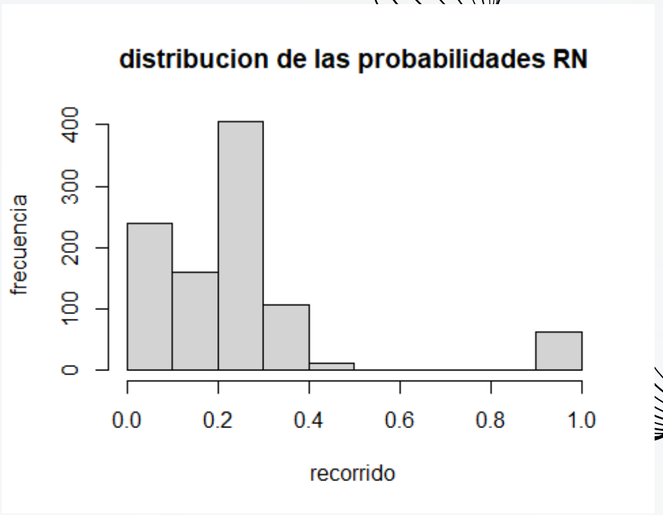


loss: 0.3886 - accuracy: 0.8370 -
val_loss: 0.4074 - val_accuracy: 0.8262

Procedimiento de discriminacion del modelo

Código de R:

```
prob_2RN = mod_RN2 %>%
  predict(as.matrix(fraude_test[,-1])) %>%
  array_reshape(., dim = c(985, 1)) %>%
  as.vector()
library(pROC)
roc_RN2=roc(as.matrix(fraude_test[,1]),prob_2RN)
plot.roc(roc_RN2,print.auc = T,print.thres = 'best',main='Curva AUC modelo RN',xlim=c(1,0))
```




```
predic_RN2=ifelse(prob_2RN>=0.253,1,0)
library(caret)
cf_RN=confusionMatrix(as.factor(predic_RN2),as.factor(as.matrix(fraude_test[,1])),positive='1')
```

Pred/Ref	0	1
0	541	22
1	195	227

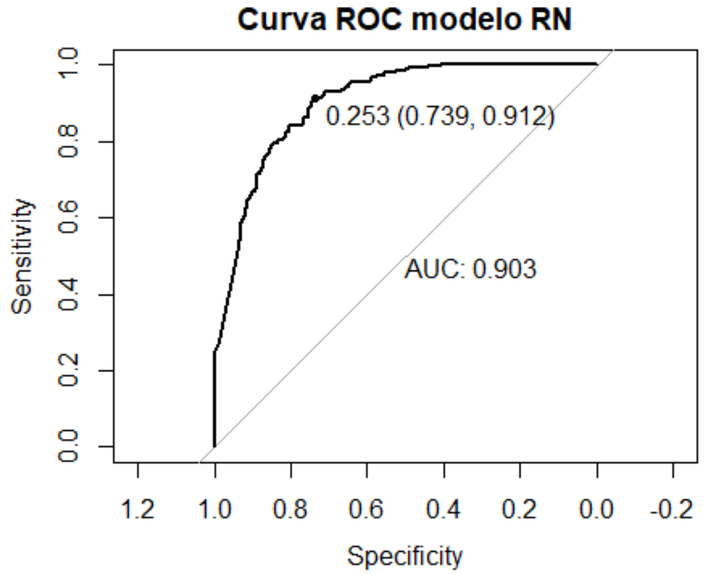
Sensibilidad:0.911
Especifidad: 0.7351

Precisión:0.537
F1 Score:0.6766

$$Precision = \frac{VP}{VP + FP}$$

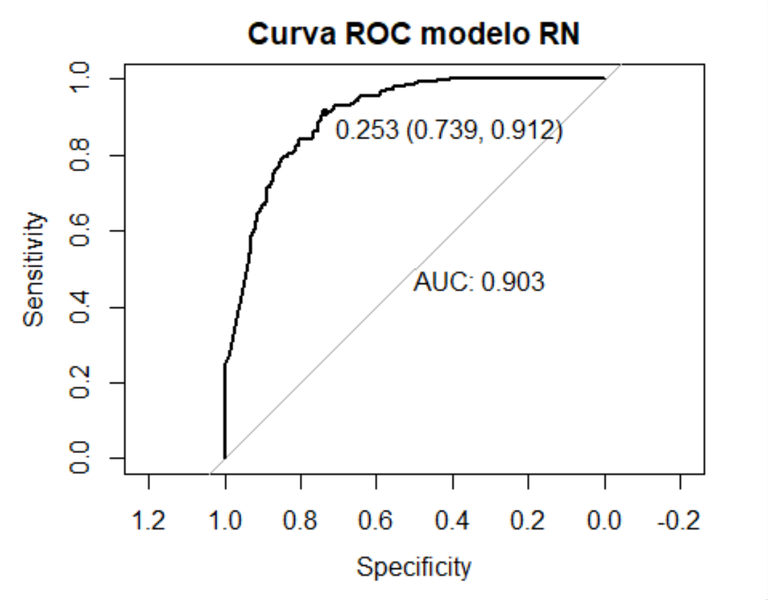
$$F1_{score} = 2\frac{Precision * S}{Presicion + S}$$

	Valores Predichos		
Valores Reales	Negativo $\hat{Y} = 0$	Positivo $\hat{Y} = 1$	Totales
Negativo Y=0	Verdaderos Negativo (VN)	Falsos Positivo (FP)	VN+FP
Positivo Y=1	Falso Negativo (FN)	Verdadero Positivo (VP)	FN+VP



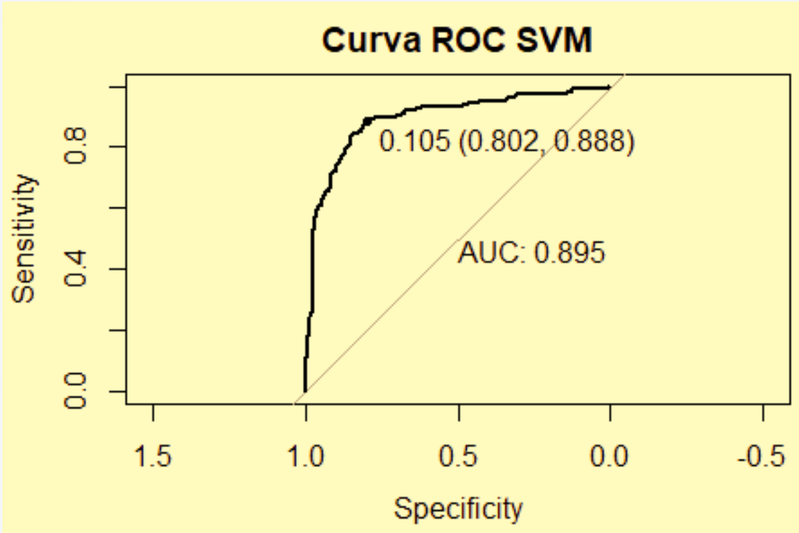
Validación de Modelos

Redes Neuronales



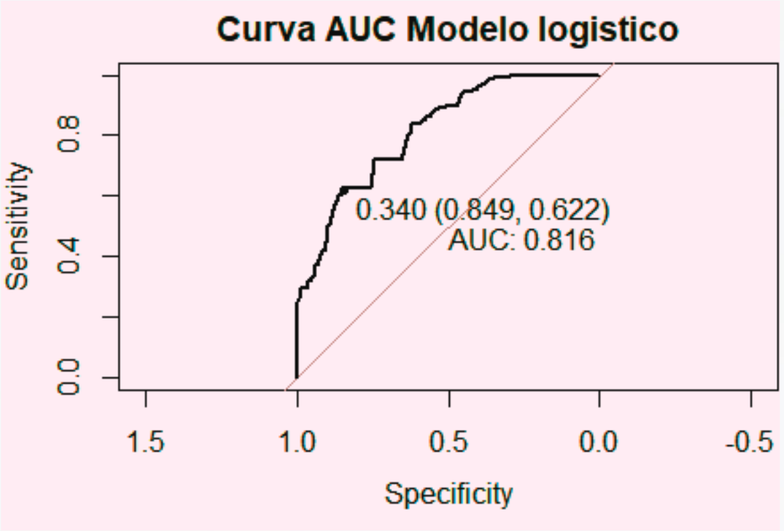
Pred/Ref	0	1
0	**541**	22
1	195	**227**

SVM



Pred/Ref	0	1
0	**590**	**29**
1	**146**	**220**

Logístico



Pred/Ref	0	1
0	**625**	**94**
1	**111**	**155**

Validación de Modelos

Redes Neuronales

Pred/Ref	0	1
0	**541**	22
1	195	**227**

Sensibilidad:0.911
Especifidad: 0.7351

Precisión:0.537
F1 Score:0.6766

SVM

Pred/Ref	0	1
0	**590**	**29**
1	**146**	**220**

Sensibilidad: 0.888
Especifidad: 0.802

Precisión: 0.601
F1 Score: 0.715

Logístico

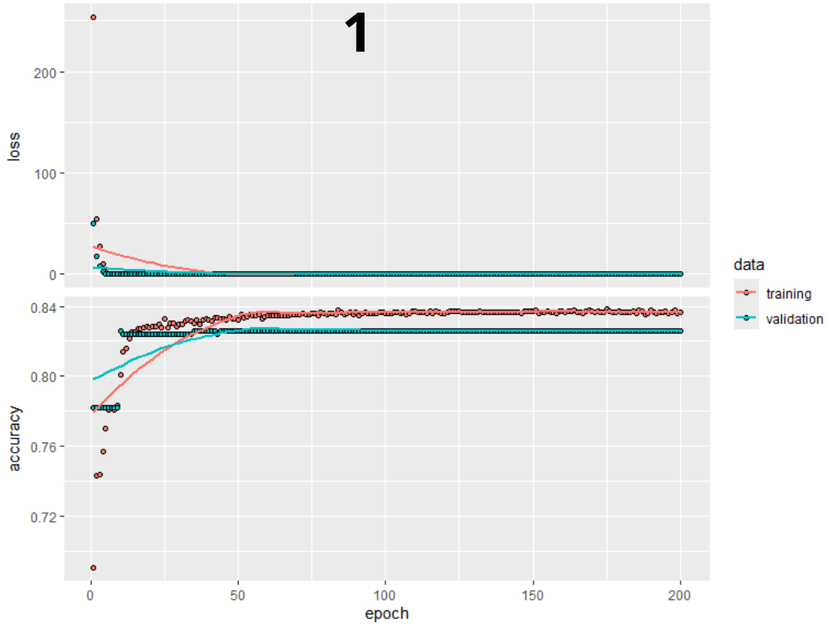
Pred/Ref	0	1
0	**625**	**94**
1	**111**	**155**

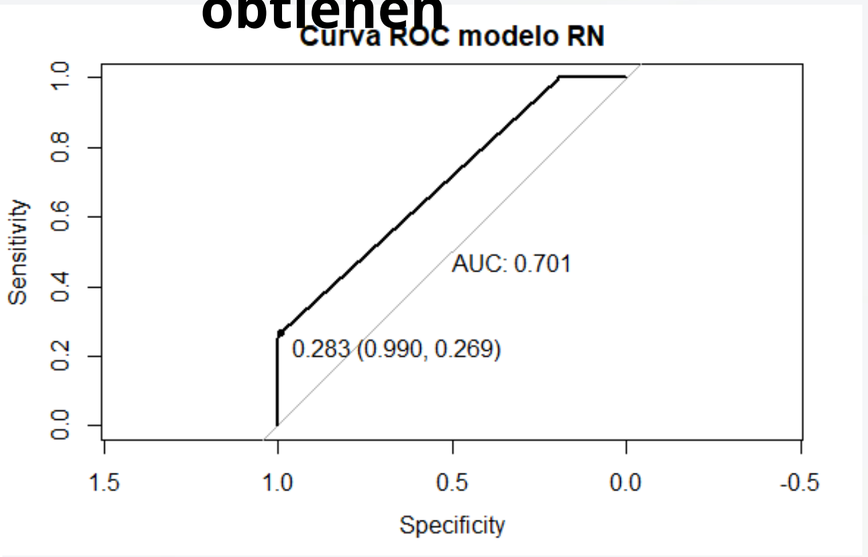
Sensibilidad: 0.622
Especifidad: 0.849

Precisión: 0.582
F1 Score: 0.601

Curva roc de las predicciones de modelos ajustados que usualmente se obtienen

Ajuste modelo habitual 1



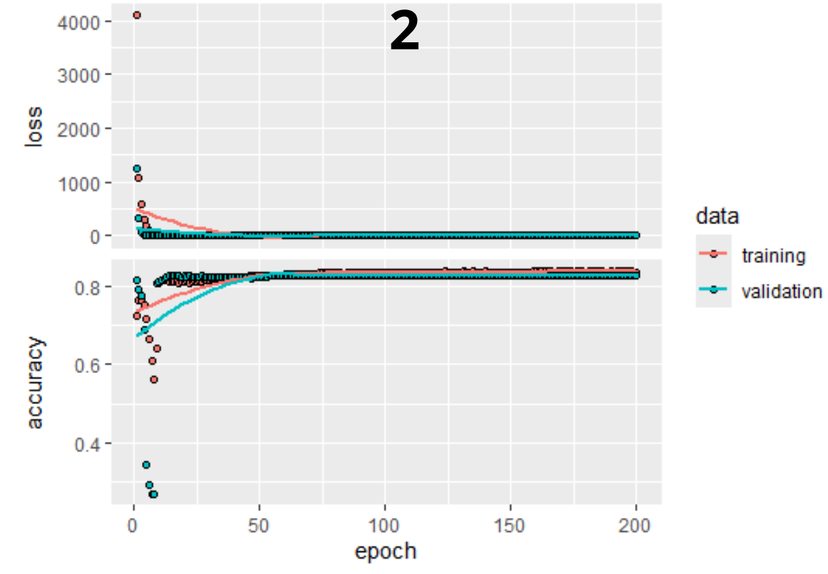


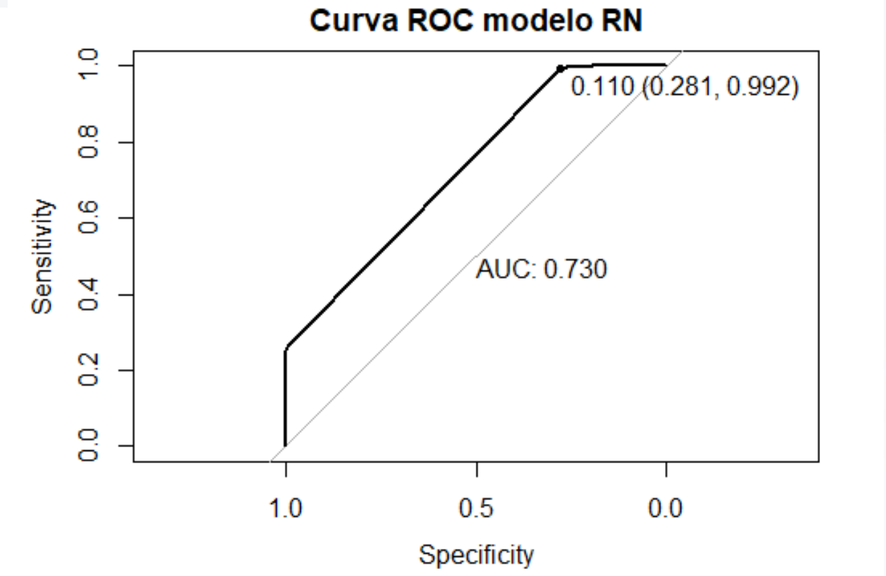
Pred/Ref	0	1
0	729	182
1	7	67

Sensibilidad:0.269
Especifidad: 0.990

Precisión:0.9054
F1 Score:0.4148

Ajuste modelo habitual 2





Pred/Ref	0	1
0	209	2
1	529	247

Sensibilidad:0.992
Especifidad: 0.281

Precisión:0.3182
F1 Score:0.4819

Estado del Arte

- **Developing a Credit Card Fraud Detection Model using Machine Learning Approaches**

Modelo logístico

F1 Score: 0.722

Pred/ Ref	0	1
0	284271	44
1	189	303

SVM

F1 Score: 0.826

Pred/ Ref	0	1
0	284261	54
1	108	384

Redes Neuronales

F1 Score: 0.763

Pred/ Ref	0	1
0	284203	112
1	119	373

LGBM: a machine learning approach for Ethereum fraud detection

Modelo logístico

F1 Score: 0.660
Precisión: 0.923

SVM

F1 Score: 0.557
Precisión: 0.994

Redes Neuronales

F1 Score: 0.883
Precisión: 0.922

Exploring the Application of Neural Network in Detecting Fraudulent within the Ethereum Network

Ref/ pred	0	1
0	1477	86
1	42	363

SVM

Sensibilidad: 0.8085
Especificidad:0.9724

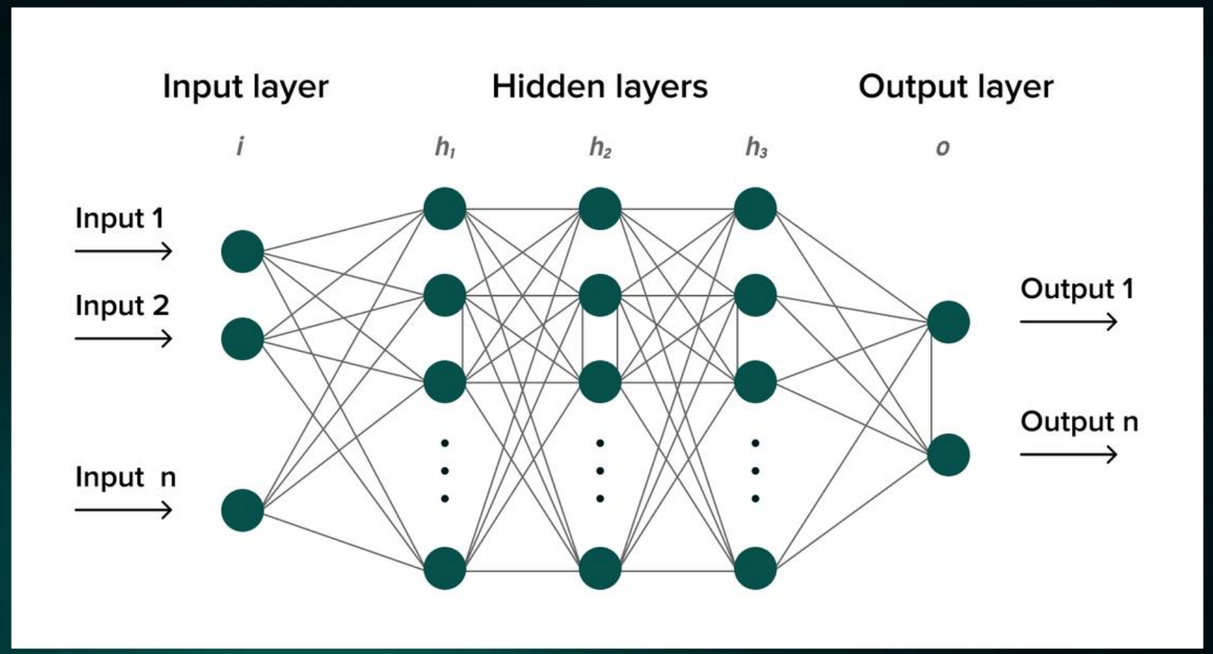
Redes Neuronales

Ref/ pred	0	1
0	1503	306
1	16	143

Sensibilidad: 0.3185
Especificidad:0.9895

Conclusión:

- Objetivos
- Hipotesis
- La importancia del falso negativo
- Mejoras : Sin limitaciones de capas
- Modelo de random Forest.



	Valores Predichos		
Valores Reales	Negativo $\hat{Y}=0$	Positivo $\hat{Y}=1$	Totales
Negativo Y=0	Verdaderos Negativo (VN)	Falsos Positivo (FP)	VN+FP
Positivo Y=1	Falso Negativo (FN)	Verdadero Positivo (VP)	FN+VP