

Data Science

Evaluación de Riesgo Crediticio

Francisco Rava

Martina Barreto Neuendorf

Agenda

- Introducción
- Objetivo del modelo
- Fuente de datos y data set
- Data Acquisition
- Data Wrangling
- Análisis exploratorio
- Modelos



Introducción

Presentaremos un modelo de machine learning para poder predecir, en base a una muestra de clientes que solicitaron prestamos al banco, la calidad crediticia de cada prestamo, mediante un resultado dicotómico. Intentaremos predecir si el crédito a otorgar será o no riesgoso.



Objetivo del modelo

Exploraremos el comportamiento de las variables de los clientes del banco alemán al momento de pedir un crédito, donde buscaremos responder preguntas como:

- ¿Con qué propósito solicitan los créditos en el banco?
- ¿Qué patrones determinan si el préstamo tendrá un buen o mal riesgo?
- ¿Quiénes son los que piden préstamos (edad, sexo, que trabajo tienen, etc)?

Objetivo de Análisis:

Minimizar la pérdida desde la perspectiva del banco, el banco necesita una regla de decisión con respecto a quién aprobar el préstamo y quién no. Dada la perspectiva del negocio, realizaremos un análisis de los datos de muestra en pos de encontrar patrones que nos ayuden a determinar si un crédito podría ser de alto o bajo riesgo.

Métricas de performance

- Al banco le resulta mas costoso otorgar créditos que erróneamente estimamos como de bajo riesgo, y que finalmente no lo sean (los falsos positivos).
- También consideramos una perdida de oportunidad y dinero subestimar los créditos no riesgosos , es decir, no darle la importancia suficiente a esa métrica. Sin embargo, concluimos que ante la necesidad de priorizar una, la primera métrica resulta mas ponderante en el negocio.

Definimos que nos va a interesar la métrica de precisión y recall, por lo tanto vamos a utilizar el f1 por promedio ponderado que pondera ambas métricas teniendo en cuenta el peso de los créditos "Good" y "Bad".

Fuente de datos

El data set cuenta con nueve variables explicativas de la variable "target", es decir, la variable la cual queremos que nuestro modelo prediga, que en este caso es "Risk".

VARIABLES:

- Age (numérica)
- Sex (male, female)
- Job (Trabajo: 0 - no calificado y no residente, 1 - no calificado y residente, 2 - calificado, 3 - altamente calificado)
- Housing (tipo de vivienda: own, rent, or free)
- Saving accounts (Caja de ahorro - little, moderate, quite rich, rich)
- Checking account (Cuenta corriente - little, moderate, rich)
- Credit amount (Monto de crédito en euros)
- Duration (Duración del crédito en meses)
- Purpose (Propósito del crédito: car, furniture/equipment, radio/TV, domestic appliances, repairs, education, business, vacation/others)
- Risk (categórica: good,bad)

Fuente: <https://www.kaggle.com/datasets/kabure/german-credit-data-with-risk>

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	49	male	1	own	little	NaN	2096	12	education	good
3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	53	male	2	free	little	little	4870	24	car	bad

Data Acquisition

#Lectura DataSet

```
df = pd.read_csv('german_credit_data.csv', sep = ',', index_col="Unnamed: 0")
df.head()
```

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	male	2	own	NaN	little	1169	6	radio/TV	good
1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	49	male	1	own	little	NaN	2096	12	education	good
3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	53	male	2	free	little	little	4870	24	car	bad

#Información adicional sobre tipos de variables que tenemos y la cantidad de registros por columna
df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Age             1000 non-null  int64
1   Sex             1000 non-null  object
2   Job             1000 non-null  int64
3   Housing         1000 non-null  object
4   Saving accounts  817 non-null   object
5   Checking account 606 non-null   object
6   Credit amount    1000 non-null  int64
7   Duration        1000 non-null  int64
8   Purpose         1000 non-null  object
9   Risk            1000 non-null  object
```

#Tamaño del data set
df.shape

(1000, 10)

```
# Vemos las variables de nuestras categóricas
unique_housing = df['Housing'].unique()
unique_purpose = df['Purpose'].unique()
unique_savings = df['Saving accounts'].unique()
unique_checking = df['Checking account'].unique()
print(unique_housing)
print(unique_purpose)
print(unique_savings)
print(unique_checking)

['own' 'free' 'rent']
['radio/TV' 'education' 'furniture/equipment' 'car' 'business'
 'domestic appliances' 'repairs' 'vacation/others']
[nan 'little' 'quite rich' 'rich' 'moderate']
['little' 'moderate' nan 'rich']
```


Data Wrangling

Tipos de variables

```
#Tipos de datos  
df.dtypes
```

Age	int64
Sex	object
Job	int64
Housing	object
Saving accounts	object
Checking account	object
Credit amount	int64
Duration	int64
Purpose	object
Risk	object

Valores nulos y su tratamiento

```
get_na(df)
```

	datos sin NAs en q	Na en q	Na en %
Checking account	606	394	39.4
Saving accounts	817	183	18.3
Age	1000	0	0.0
Sex	1000	0	0.0
Job	1000	0	0.0
Housing	1000	0	0.0
Credit amount	1000	0	0.0
Duration	1000	0	0.0
Purpose	1000	0	0.0
Risk	1000	0	0.0

```
df["Saving accounts"].fillna("No",inplace=True)  
df["Checking account"].fillna("No",inplace=True)  
df.head(5)
```

Reemplazamos los NaN por 'No' para Checking y Savings accounts ya que interpretamos que se pusieron valores nulos porque no poseían el tipo de # cuenta de la columna

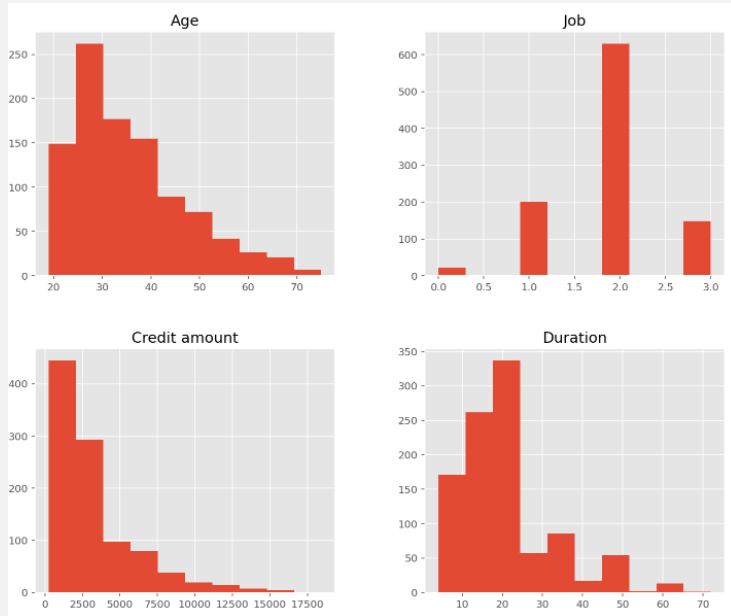
	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose	Risk
0	67	male	2	own	No	little	1169	6	radio/TV	good
1	22	female	2	own	little	moderate	5951	48	radio/TV	bad
2	49	male	1	own	little	No	2096	12	education	good
3	45	male	2	free	little	little	7882	42	furniture/equipment	good
4	53	male	2	free	little	little	4870	24	car	bad



Análisis Exploratorio

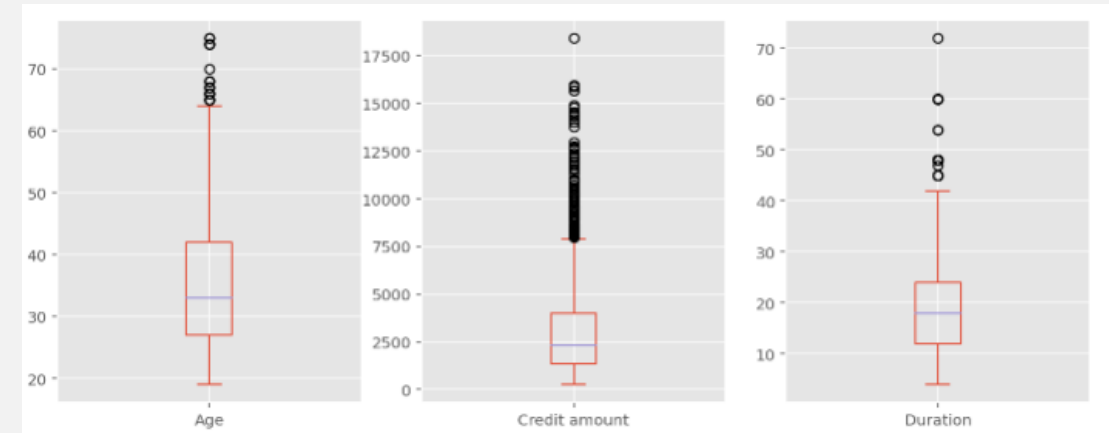
Buscamos las relaciones entre las
variables explicativas y target.

Análisis Univariado

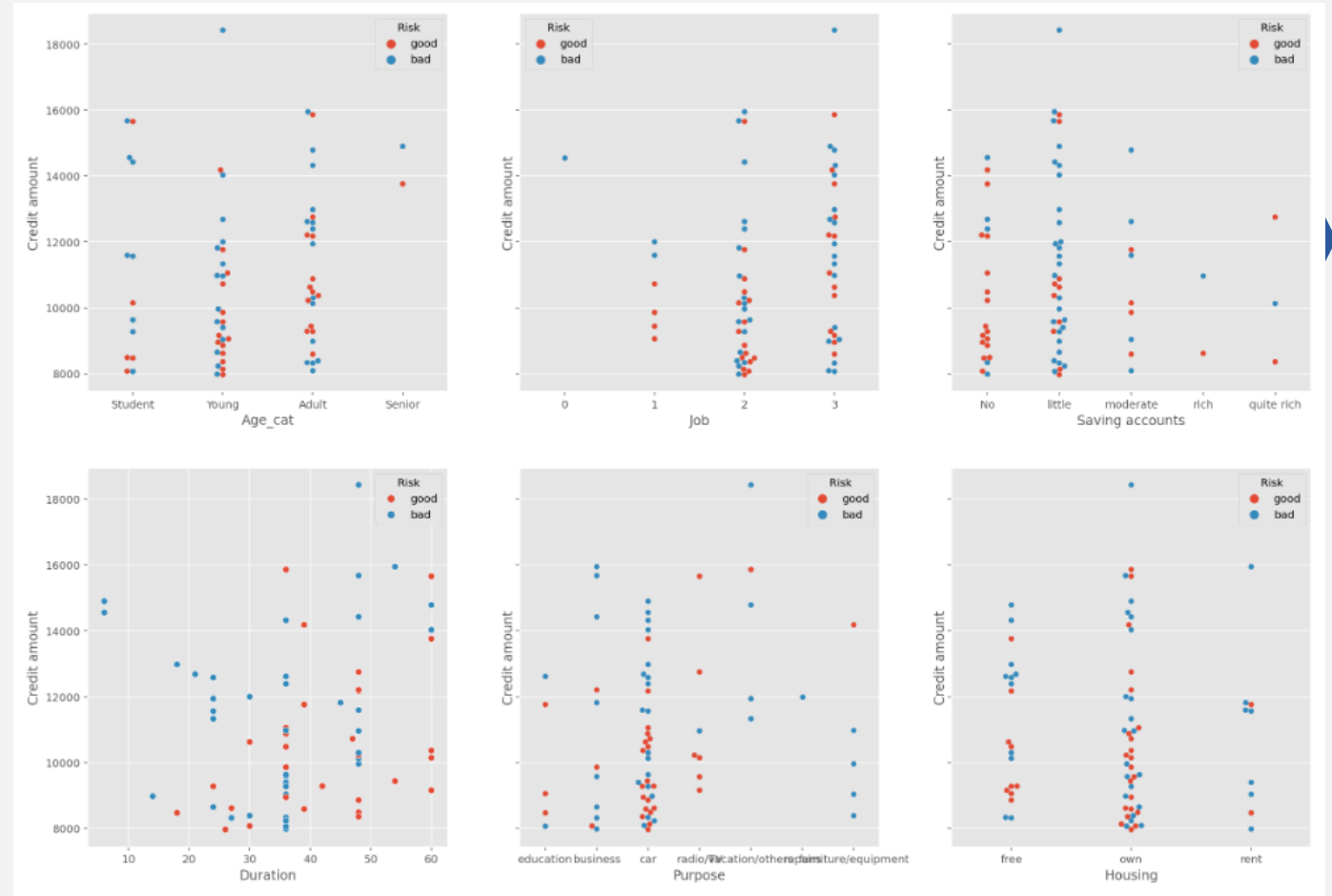
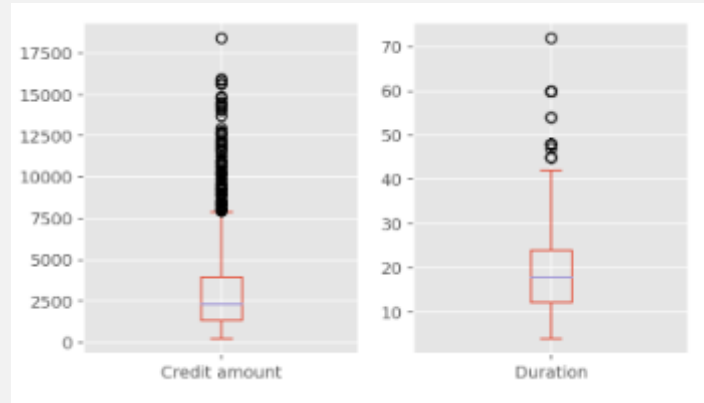


- Podemos ver que los que mas piden prestamos son los que tienen trabajos de tipo 2 : Calificado
- Las edades con un pico de frecuencia son entre los 25 y 30 años, estos son los mas probables a la hora de pedir un credito.
- El 63% de los trabajos son de tipo 2 (Calificado) y el 83% de los datos de trabajo los acumulan los de tipo 2 y 1 (No calificado y residente)
- Gran parte de los créditos son repagados a los 20-25 meses de su otorgamiento.

- No podemos tener en cuenta la media como representativa porque los datos tiene un % grande de desviación estandar respecto a ella
- Credit Amount tiene muchos valores outliers que podemos ver en el grafico y además vemos que el 75% de los créditos son de hasta 4000 euros
- En duration el 75% de los créditos son de hasta 24 meses, con un máximo de 72m.
- En Age, el 50% de las personas tienen entre 27 y 42 años , y no presenta demasiados outliers

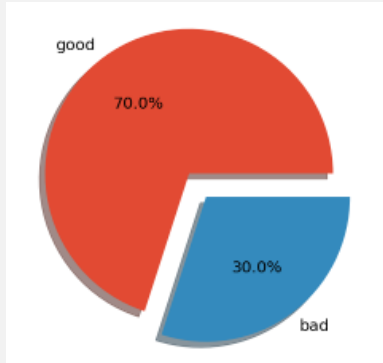


Análisis de Outliers

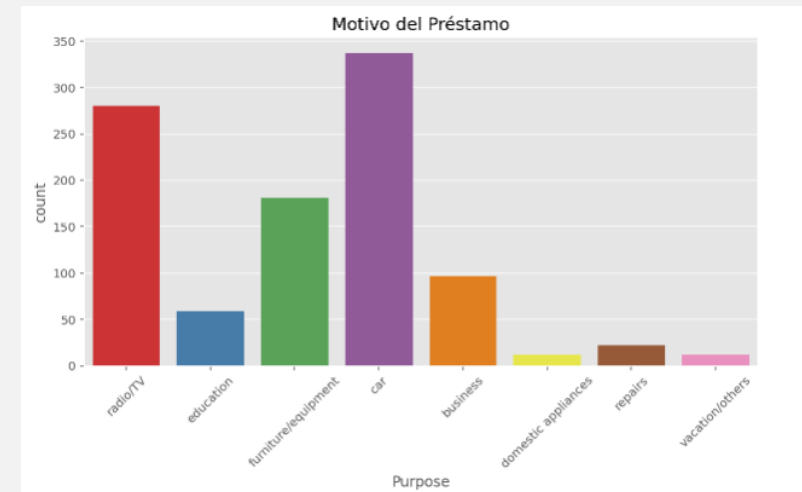


Análisis Univariado

Variable Target
"Risk"

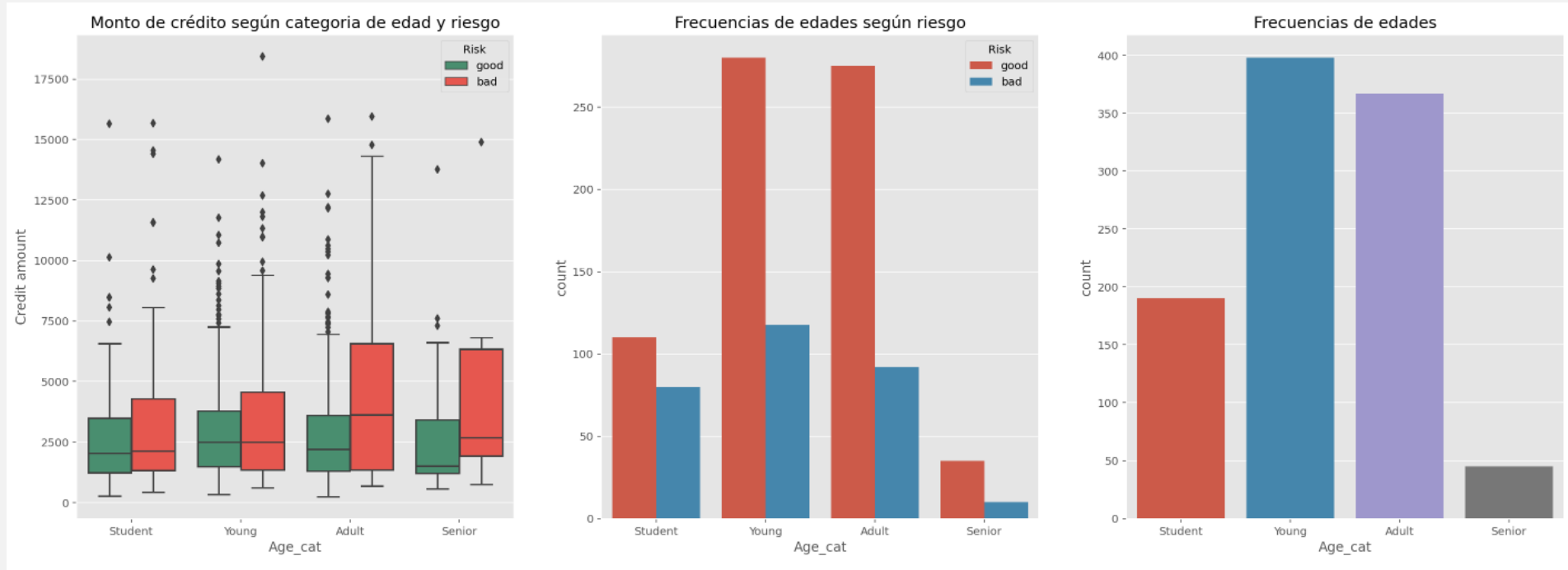


- Target --> vemos que tenemos mucho mas frecuencia de buenos créditos, tendremos que balancear esta variable
- Checking account --> vemos que la mayoría no tiene cuentas de cheque y de los que sí tienen cuenta, tenemos iguales cantidad de leves y moderados, y tenemos mucha menos frecuencia en rich
- Saving accounts --> gran diferencia entre las saving account de bajo monto en relación al resto de las categorías
- Motivo de préstamo --> la mas frecuente es para auto y le sigue para radio/tv (ambas se diferencian bastante del resto)
- Housing --> gran diferencia de frecuencias, la mayoría de la muestra tienen casa propia



Análisis Bivariado y Multivariado

“Age”

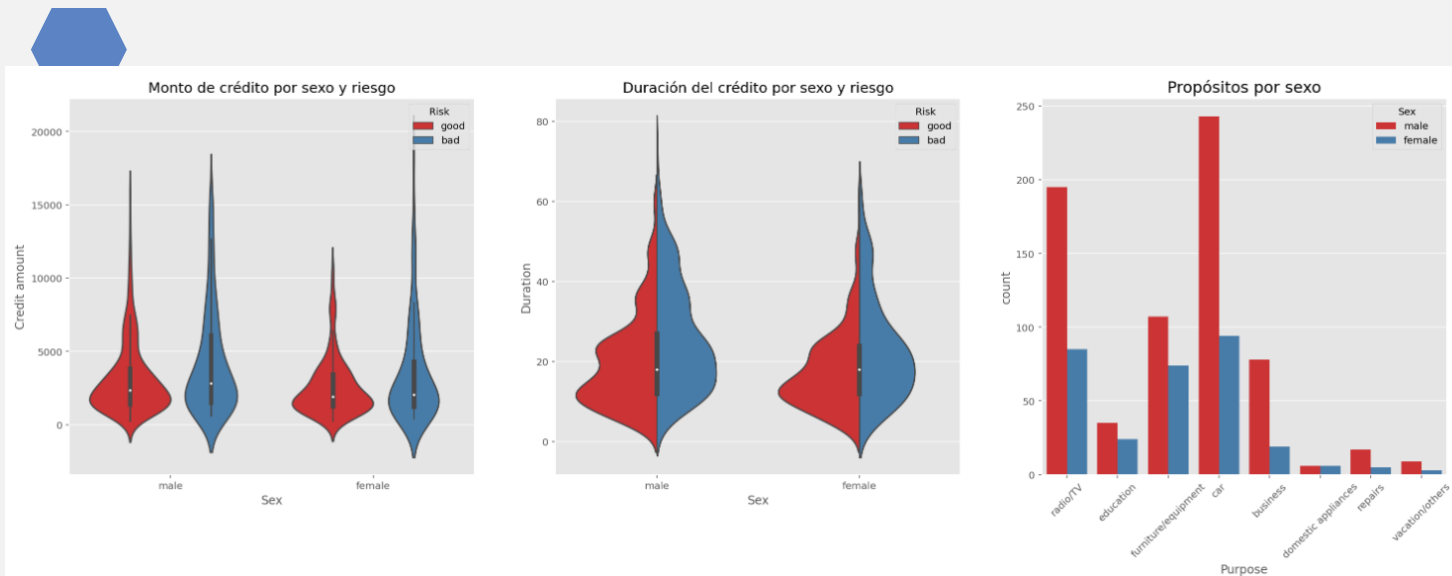
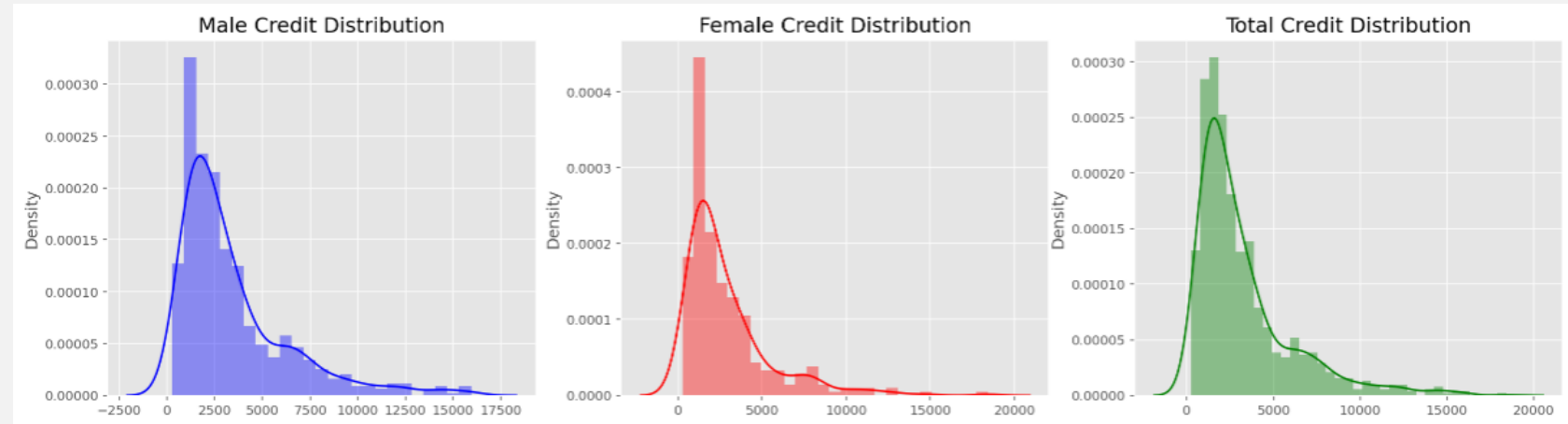


- Podemos ver que son muchos mas los Young y Adult que piden créditos en comparación con los Senior y Student
- Viendo las variaciones que tiene cada categoría entre los riesgos, vemos que hay una gran diferencia de riesgos intragrupal de Young y Adult. Por que se puede dar que en un mismo rango de edades varien tanto sus riesgos?
- Podemos ver que hay una clara relación, para los grupos de Adult y Senior, entre monto de crédito y cuan riesgoso es. Vemos que los montos mas grandes de crédito son créditos de alto riesgo para estos grupos.

Análisis Bivariado y Multivariado

“Sex”

Eliminamos la variable sex del modelo porque no aporta información acerca de la variable target y suele no estar permitida en los análisis crediticios.



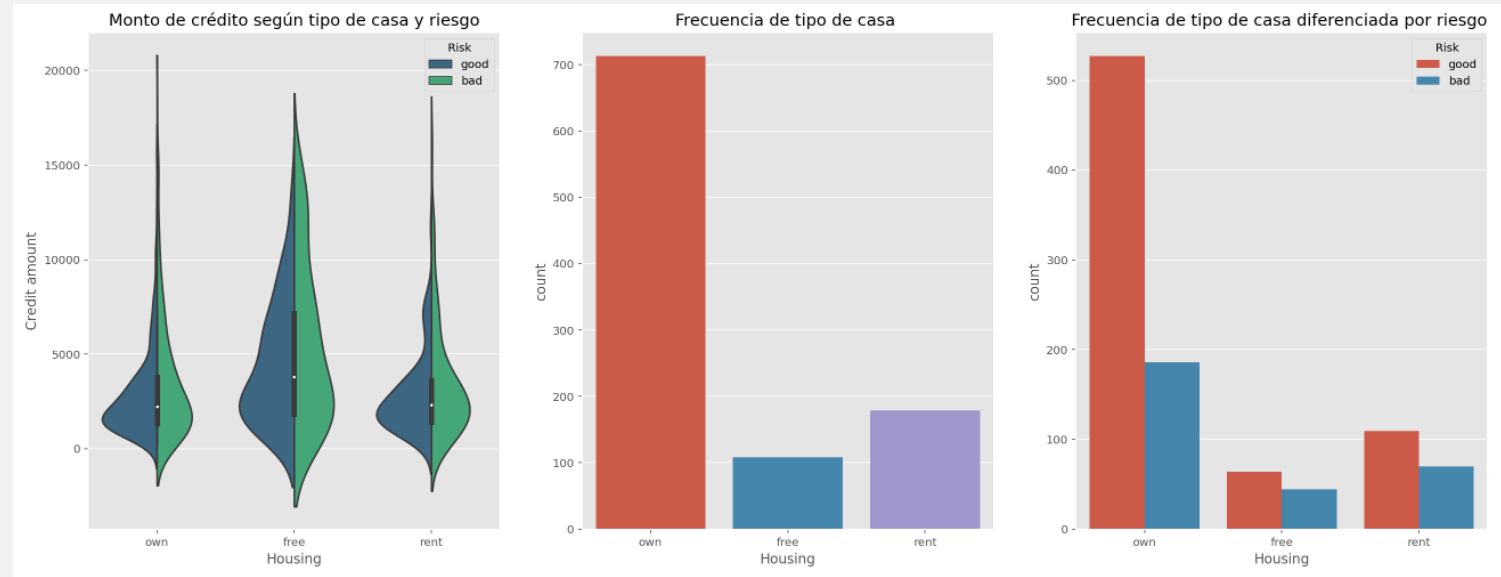
- La distribución de la variable sex según el riesgo, es la misma para credit amount y duration.
- En proporción los hombres y las mujeres piden créditos por los mismos propósitos
- No hay un sexo que sea significativamente mas riesgoso que el otro.

Análisis Bivariado y Multivariado

"Housing"

Credit amount y Housing

- Vemos que con el tipo de casa no hay diferencias marcadas en la distribución, para los que tienen casa propia notamos una mayor concentración entre los 1000 y 2000 euros, aunque recordar que "own" concentra el 72% de los datos, después notamos que son datos bastante dispersos.
- Mayor concentración de riesgos crediticios malos para personas con casa gratuita pero no propia entre los 1500 euros aprox, en relación a los riesgos malos.
- Apenas mayor concentración de riesgos buenos en clientes con casa propia para bajos montos (1500 aprox) en comparación con los riesgos altos que están más dispersos en esta variable.

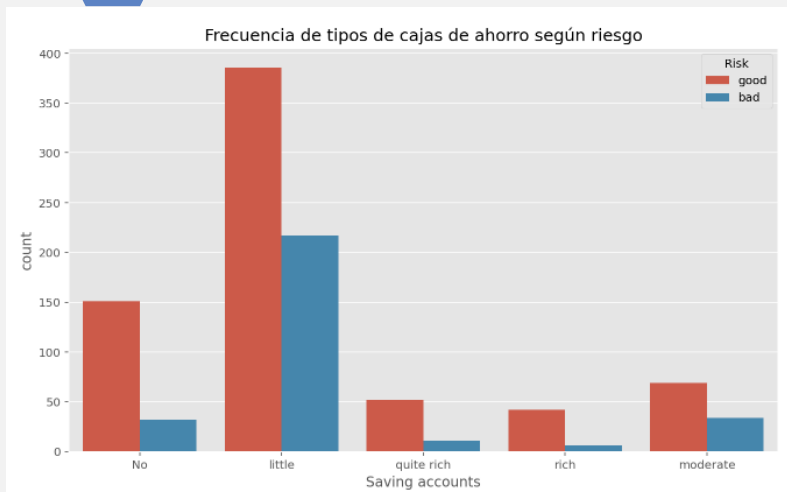
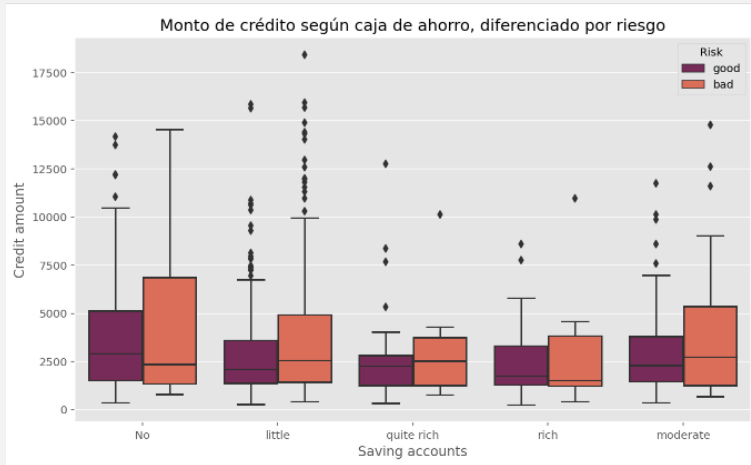


Frecuencia de Housing

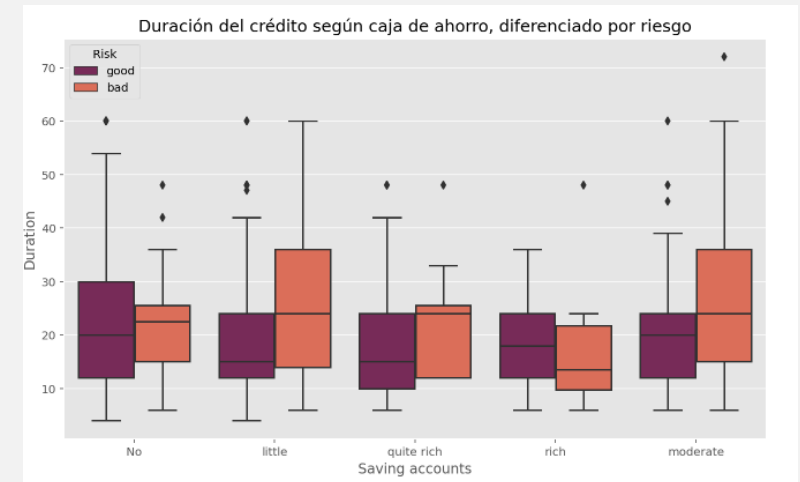
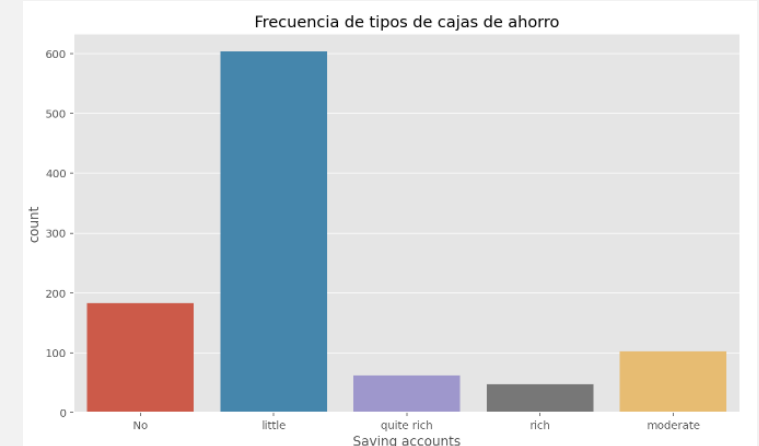
- La mayoría de los aplicantes tienen casa propia ("own") y con el tercer gráfico notamos que los que tienen casa propia si generan una diferencia en el resultado crediticio. Es decir, más de la mitad de los que poseen una casa (72% aprox), son calificados como no riesgosos. Este tipo de diferenciación en los riesgos no es tan notoria en los demás tipos de casa. Además notamos que los que tienen casa propia y los que rentan, son más propicios a pedir montos bajos de crédito, que los que tienen casa gratuita, cuyos montos de créditos son mucho más dispersos tanto con riesgos buenos como malos.

Análisis Bivariado y Multivariado

"Savings Account"

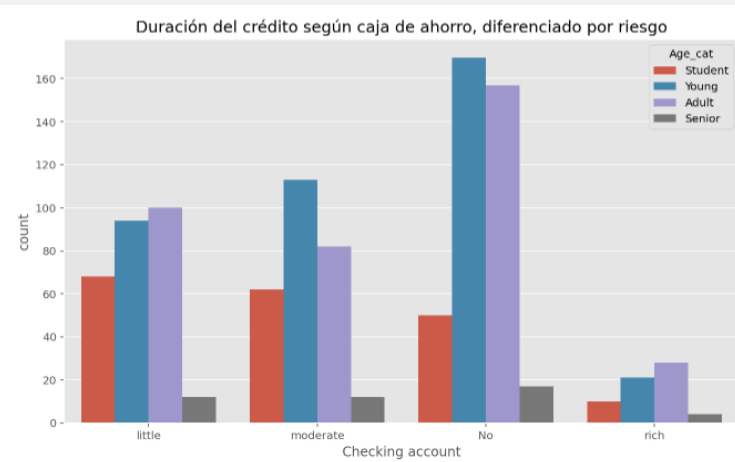
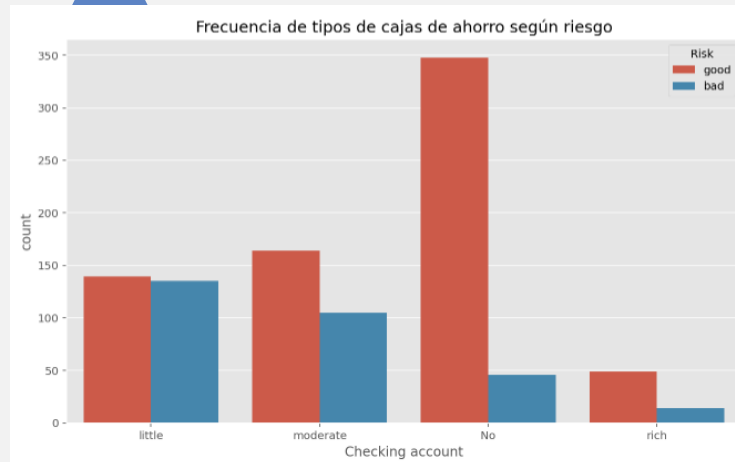
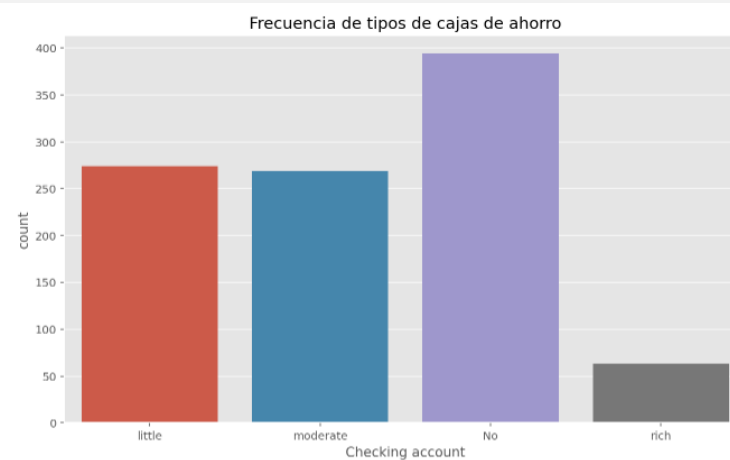
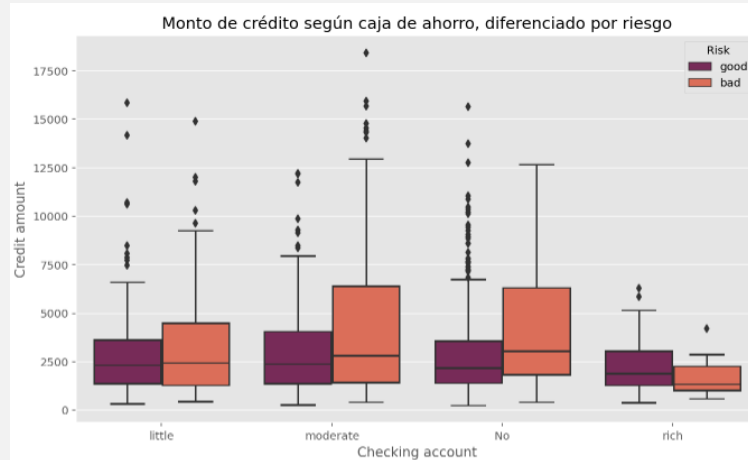


- Con duraciones mayores a los 30 meses son mas probables de ser clasificados como créditos riesgosos.
- La mayoría de los deudores tienen poco monto o en segundo lugar no tienen cajas de ahorro ("little" o "no").
- Los que tienen cuentas de ahorro con mas monto, son mas propensos a ser calificados de bajo riesgo ("rich" y "quite rich").
- Los que tienen cajas de ahorro de tipo "no", "little" y "moderate" son mas probables de ser clasificados como riesgosos si superan el monto de los 5000 euros.




Análisis Bivariado y Multivariado

“Checking account”



- La mayoría de los aplicantes no tienen cuenta corriente.
- Los que no tienen cuenta corriente son ampliamente categorizados como de bajo riesgo.
- Mas de la mitad de los que tienen cuentas corrientes de montos acaudalados (rich) son calificados como de bajo riesgo.
- Mas del 50% de los que no tienen cuenta corriente son jóvenes (Young: 25 - 35 años) y adultos (Adult: 35 - 60 años).
- Los aplicantes con cuentas corrientes de poco monto o que no tienen cuenta ("no", "little" and "moderate") que a su vez piden montos mas altos de crédito tienen mas probabilidad de ser clasificados como riesgosos.



Modelos de predicción

Modelos de Clasificación – Preparación del data set

```
# Transformo las variables objeto en categoricas para que pueda correr el modelo
df["Housing"] = df["Housing"].astype('category')
df["Saving accounts"] = df["Saving accounts"].astype('category')
df["Checking account"] = df["Checking account"].astype('category')
df["Purpose"] = df["Purpose"].astype('category')
df["Age_cat"] = df["Age_cat"].astype('category')
df["Risk"] = df["Risk"].astype('category')
```

Genero el encoder

```
oe_style = OneHotEncoder() #Para variables que tienen dos valores unicos es mejor transformarlas a binarias
oe_results = oe_style.fit_transform(df[["Risk"]])
df[["good", "bad"]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
oe_results = oe_style.fit_transform(df[["Age_cat"]])
df[["Senior", "Student", "Adult", "Young"]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
oe_results = oe_style.fit_transform(df[["Housing"]])
df[["own", "free", "rent"]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
oe_results = oe_style.fit_transform(df[["Saving accounts"]])
df[["No", "little", "quite rich", "rich", "moderate"]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
oe_results = oe_style.fit_transform(df[["Checking account"]])
df[["little", "moderate", "No", "rich"]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
oe_results = oe_style.fit_transform(df[["Purpose"]])
df[["radio/TV", "education", "furniture/equipment", "car", "business",
     "domestic appliances", "repairs", "vacation/others"]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
oe_results = oe_style.fit_transform(df[["Job"]])

df[[2, 1, 3, 0]] = pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_)
```

Eliminamos las variables que quedaron de mas

```
X = df.drop(['Sex', 'Risk', "good", "bad", "Housing", "Saving accounts",
            "Checking account", "Purpose", "Age_cat", "Job"], axis=1) #Elimino de mi dataset la variable a predecir y las que estan de mas
y = df.good #Defino el Target
```

Escalamiento de las variables

#Escalamos las variables para que queden en una misma unidad de medida

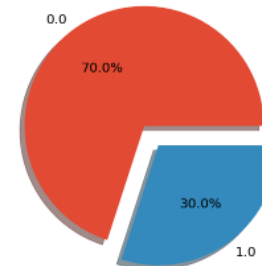
```
X.loc[:, 'Credit amount'] = MinMaxScaler().fit_transform(X[['Credit amount']])
X.loc[:, 'Duration'] = MinMaxScaler().fit_transform(X[['Duration']])
```

Separo el data set: 30% Test y 70% Train

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=5, stratify=y)
```

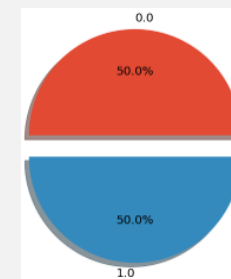
La variable target esta desbalanceada

plot_pie(y_train)



Balanceo las clases

```
# El balanceo se puede hacer con under sampler o over sample
# Eligo under, ya que es el que menos valores sinteticos crea de los dos
undersample = RandomUnderSampler(
    sampling_strategy='majority', random_state=0).fit(X_train, y_train)
X_train_us, y_train_us = undersample.fit_resample(X, y)
print('Información del dataset con Random Under Sampling:')
      '\n y: {}'.format(Counter(y_train_us)))
plot_pie(y_train_us)
```



Modelos de Clasificación – Árbol de decisión (un solo ejemplo demostrativo)

Para encontrar los mejores hiperparámetros para el modelo se uso GridSearch con profundidad de 2 y la cantidad mínima de muestras por hoja es 5.

Resultados (train vs test)

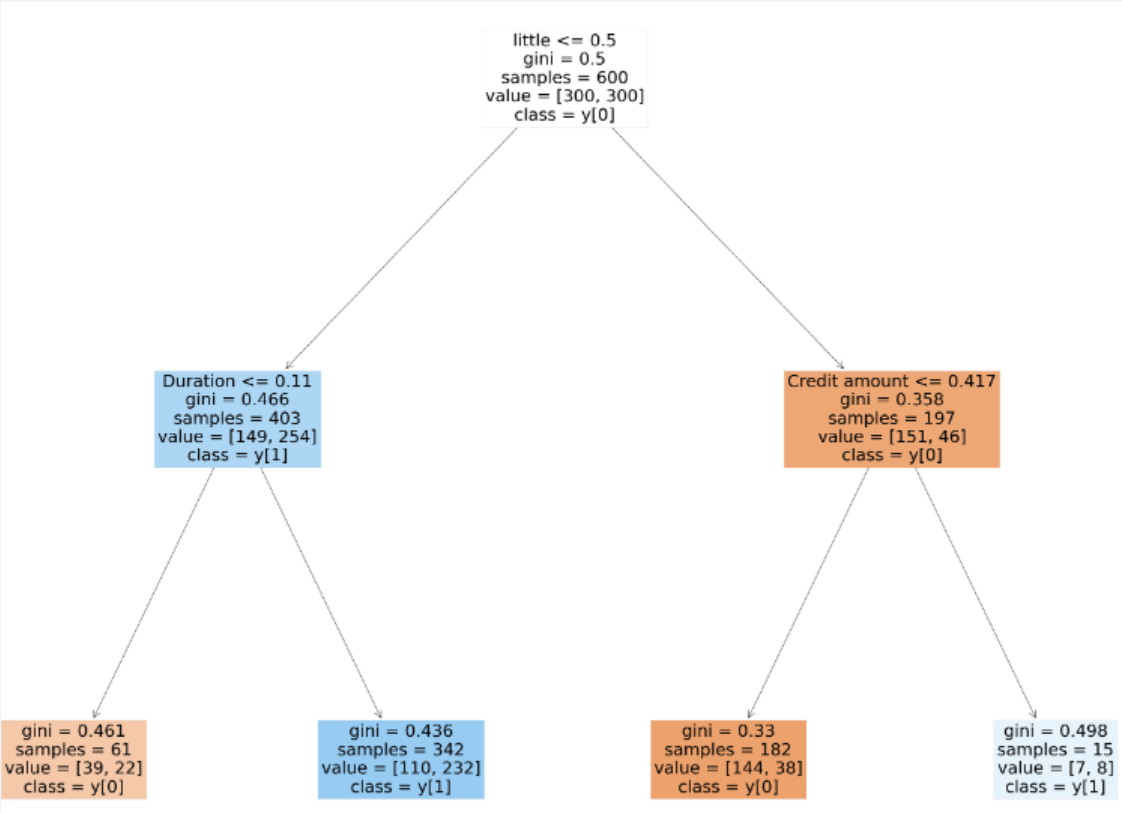
```
# Creamos el modelo
arbol_de_decision = DecisionTreeClassifier(max_depth=2, random_state = 5,min_samples_leaf=15)
# Entrenamos el modelo con el 70% de Los datos de entrenamiento
arbol_de_decision.fit(X_train_us,y_train_us)
# Hago La prediccion con Los datos de entrenamiento
prediccion = arbol_de_decision.predict(X_train)
print(classification_report(y_train,prediccion))
```

	precision	recall	f1-score	support
0.0	0.87	0.61	0.72	490
1.0	0.46	0.80	0.59	210
accuracy			0.66	700
macro avg	0.67	0.70	0.65	700
weighted avg	0.75	0.66	0.68	700

```
# Hago La prediccion con La parte de Los datos que defini como test, 30%
prediccion = arbol_de_decision.predict(X_test)
print(classification_report(y_test,prediccion))

# Bajan todas Las metricas, esto es producto de overfitting del modelo en Los datos de entrenamiento
```

	precision	recall	f1-score	support
0.0	0.87	0.54	0.67	210
1.0	0.43	0.81	0.56	90
accuracy			0.62	300
macro avg	0.65	0.68	0.62	300
weighted avg	0.74	0.62	0.64	300



```
# Al contruir el modelo con Los datos de entrenamiento, predigo con Los datos del X test y armo La matriz de confusión
# comparandolo con Y test
prediccion = arbol_de_decision.predict(X_test)
pd.crosstab(y_test,prediccion,rownames=['Actual class'], colnames=['Predicted class'])
```

Predicted class	0.0	1.0
Actual class		
0.0	114	96
1.0	17	73

Resultados



Desde el punto de vista del negocio la primer métrica me permite obtener una ganancia al estimar correctamente los créditos de riesgo "good" y la segunda métrica me permite anticiparme a un crédito "bad" y no otorgarlo, ya que me generaría una futura pérdida

Comparación entre modelos

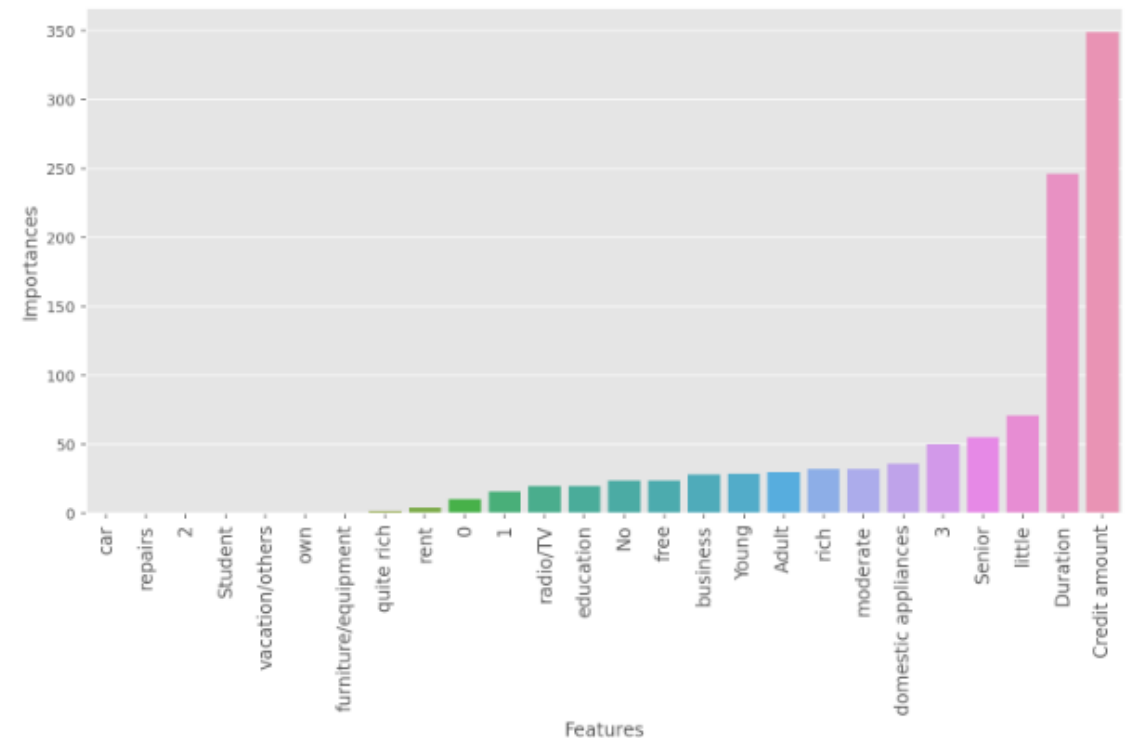
	Árbol de Decisión	Random Forest	Regresion Logística	SVM	LGBM
f1-Score	0.637	0.734	0.694	0.642	0.742
El modelo que mejor performance es LGBM					

Evaluación del modelo

Predicted class	0.0	1.0
Actual class		
0.0	160	58
1.0	23	59

- De 218 créditos que estimo que son "good", 160 son correctos (73%) y 58 no lo son (los falsos positivos)
- De 82 créditos que predigo que son "bad", 59 son correctos (72%) y 23 no lo son (los falsos negativos).

Random Forest: Importancia de cada feature



Conclusión

- Se desarrollaron diferentes modelos de Machine Learning y con el que mejores resultados se obtuvieron fue el Random Forest capaz de clasificar a los clientes del banco e identificar cuales eran riesgosos y cuales no.
- Se elimino el overfitting, de esta manera el modelo seleccionado identifica el patrón que hay en el data set y tiene la capacidad de predecir ante valores no conocidos.

