ITESO

Universidad Jesuita
de Guadalajara

MODELO REGRESIÓN LINEAL

Santiago Aguirre Vera

Ángel de la Mora

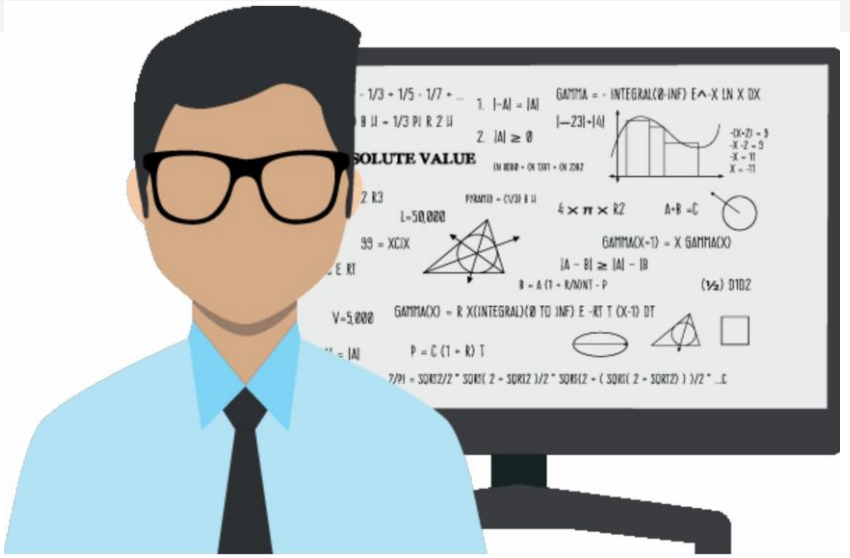
PROBLEMA A RESOLVER

- Se tiene una base de datos con 6 columnas, que nos dan información sobre datos de la persona y con la variable a predecir que será **Salary.**
- Haremos 4 modelos diferentes de regresión, y buscaremos optimizar los mejores parámetros para estos modelos y ver cuál entiende mejor nuestros datos.
- Se hará una limpieza para poder quitar datos atípicos o nulos, ver visualizaciones y poder entender nuestros datos antes de empezar el modelado.

APLICACIÓN DEL PROBLEMA

- Poder predecir el salario de una persona, mediante su experiencia trabajando, su edad, nivel de estudio y carrera, es algo que se puede utilizar de una manera frecuente, ya que nos ayudará a estimar cuanto debería ganar esa persona mediante las características que tiene y tener una idea de cuánto sería lo más correcto que tiene de salario esa persona.

Descripción de los Datos



- **Age**:This column represents the age of each employee in years. The values in this column are numeric.
- **Gender**: This column contains the gender of each employee, which can be either male or female. The values in this column are categorical.
- **Education Level**: This column contains the educaional level of each employee, which can be high school, bachelor's degree, master's degree, or PhD. The values in this column are categorical.
- **Job Ttile**: This column contains the job title of each employee. The job titles can vary depending on the company and may include positions such as manager, analyst, engineer, or administrator. The values in this column are categorical.
- **Years of experience**: This column represents the number of years of work experience of each employee. The values in this column are numeric.
- **Salary**: This column represents the annual salary of each employee in US dollars. The values in this column are numeric and can vary depending on factors such as job title, years of experience, and education level.

DATOS DATAFRAME

- **Shape** (tamaño del dataframe)

```
#tamaño de nuestro dataframe
df.shape
```

```
(375, 6)
```

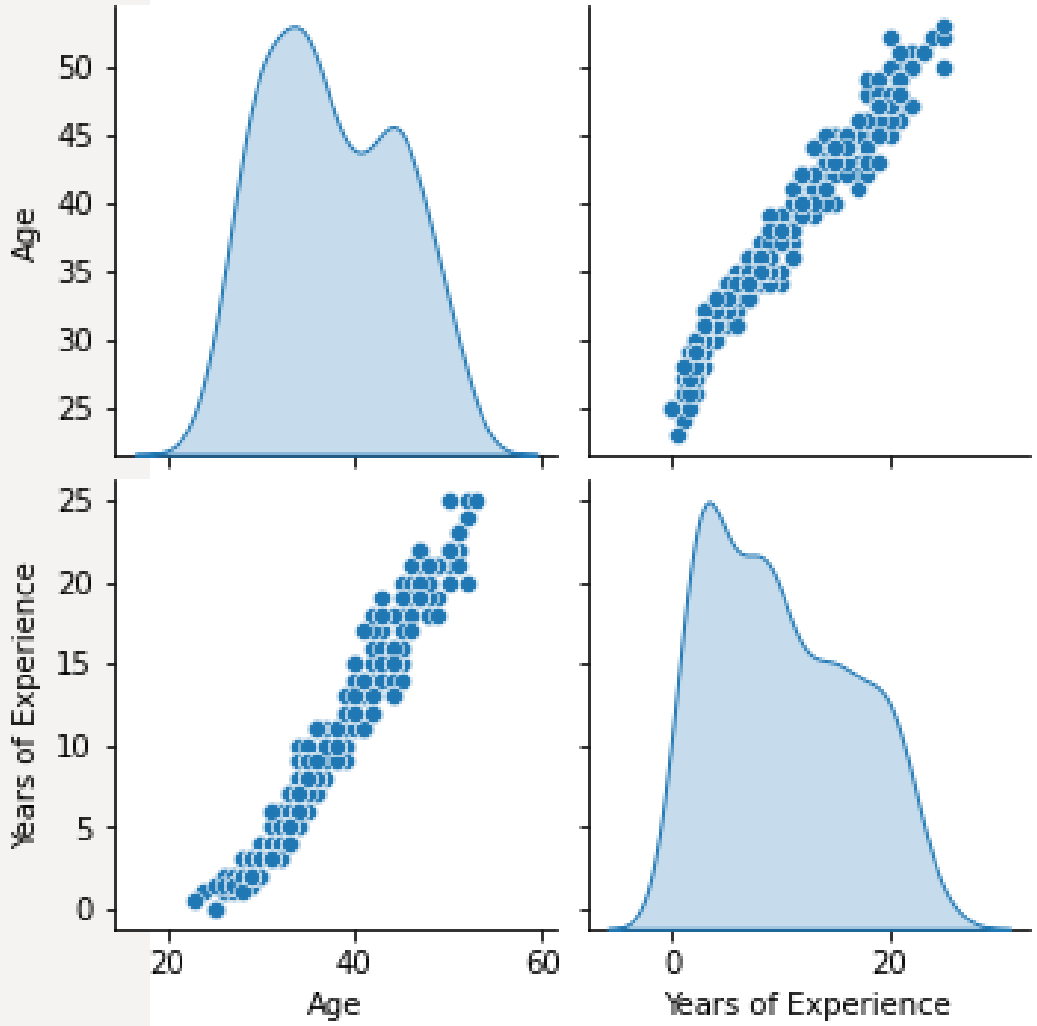
- **Dtypes**

```
Age                  float64
Gender                object
Education Level       object
Job Title             object
Years of Experience  float64
Salary               float64
dtype: object
```

CARDINALIDAD

- Vemos algo muy importante en nuestros datos que es que hay mucha cardinalidad en la variable **Job Title**, esto queda como **nota,** ya que dejaremos así los datos, pero podemos realizar mejor un área de **Job Title** y juntar los trabajos más parecidos en un solo grupo y poder dejar con menos cardinalidad nuestra variable.

```
La cardinalidad de la variable Age es 31
La cardinalidad de la variable Gender es 2
La cardinalidad de la variable Education Level es 3
La cardinalidad de la variable Job Title es 174
La cardinalidad de la variable Years of Experience es 28
La cardinalidad de la variable Salary es 33
```



VISUALIZACIÓN NUMÉRICAS

- VISUALIZAMOS **YEARS OF EXPERIENCE** Y **AGE.**

- **AGE:** Vemos una distribución bastante normal, dando a entender que nuestros datos son entre 35 años los más frecuentes y tienen una relación lineal los datos.
- **YEARS OF EXPERIENCE:** Se ve que los años de experiencia rondan los 10 años y pocos con 20 años y de igual forma de comportan de manera lineal.

DESCRIPCIÓN NUMÉRICAS

- AQUÍ VEMOS QUE LO DICHO ANTERIORMENTE TIENE CONGRUENCIA ESTADISTICAMENTE

	count	mean	std	min	25%	50%	75%	max
Age	373.0	37.431635	7.069073	23.0	31.0	36.0	44.0	53.0
Years of Experience	373.0	10.030831	6.557007	0.0	4.0	9.0	15.0	25.0

NUEVAS VARIABLES

- Creamos estas variables para poder aportar más información a nuestro dataframe y ver si esto puede ayudar a nuestro modelo a entender nuestros datos, siempre y cuando no sobreajuste los datos.

salario_anual	nivel de experiencia	Edad de inicio laboral	Nivel_salario
18000.000000	intermedio	27.0	0
21666.666667	intermedio	25.0	0
10000.000000	experimentado	30.0	1
8571.428571	intermedio	29.0	0
10000.000000	experimentado	32.0	1

CREACIÓN NUEVAS VARIABLES

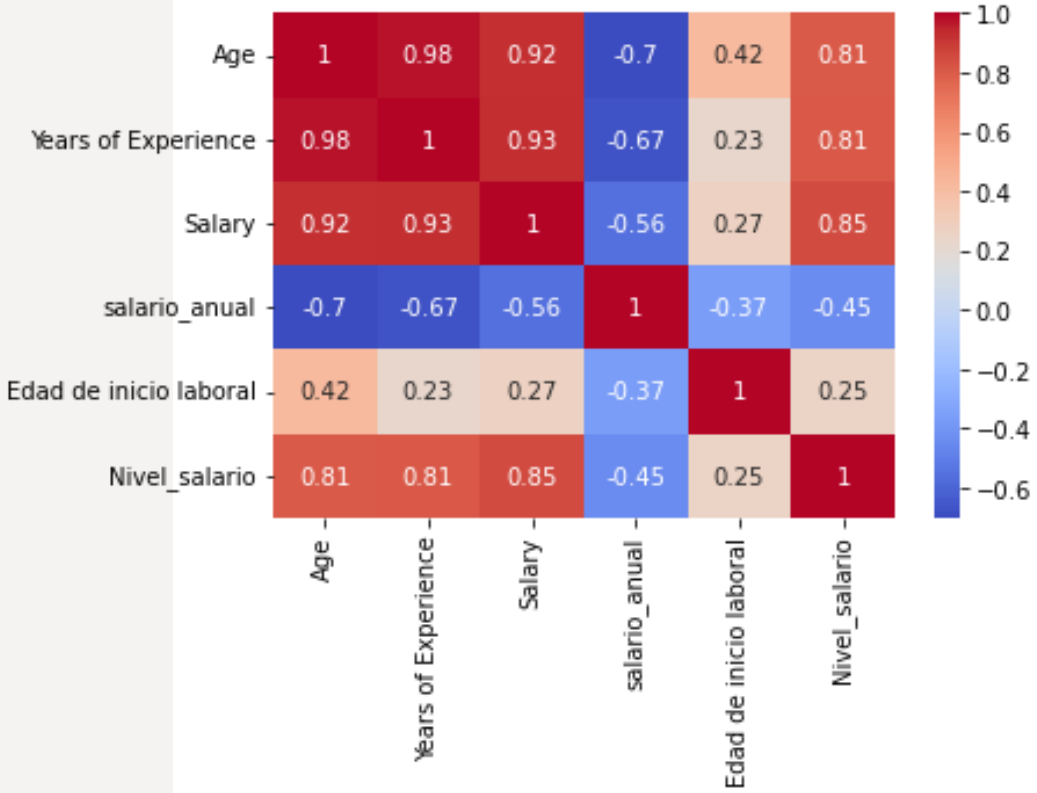
```python
def nivel_de_experiencia(años_de_experiencia):
    if años_de_experiencia < 3:
        return "principiante"
    elif años_de_experiencia >= 3 and años_de_experiencia <= 10:
        return "intermedio"
    else:
        return "experimentado"
```

```python
df2['nivel de experiencia'] = df2['Years of Experience'].apply(nivel_de_experiencia)
```

```python
df2['salario_anual'] = df['Salary'] / df2['Years of Experience']
```

```python
df2['Edad de inicio laboral'] = df2['Age'] - df2['Years of Experience']
```

```python
# calcular el promedio de salarios
avg_salary = df['Salary'].mean()

# crear la nueva columna Salary_Level basada en el promedio de salarios
df2['Nivel_salario'] = df2['Salary'].apply(lambda x: 1 if x > avg_salary else 0)
```



MATRIZ DE CORRELACIÓN

- Vemos relaciones altas positivas y también inversas con la variable respuesta.
- La variable **Edad de incio laboral** fue la más baja y está considerada para eliminarse, en caso de ser necesario (la dejamos por tener pocos datos).

ENCODEO

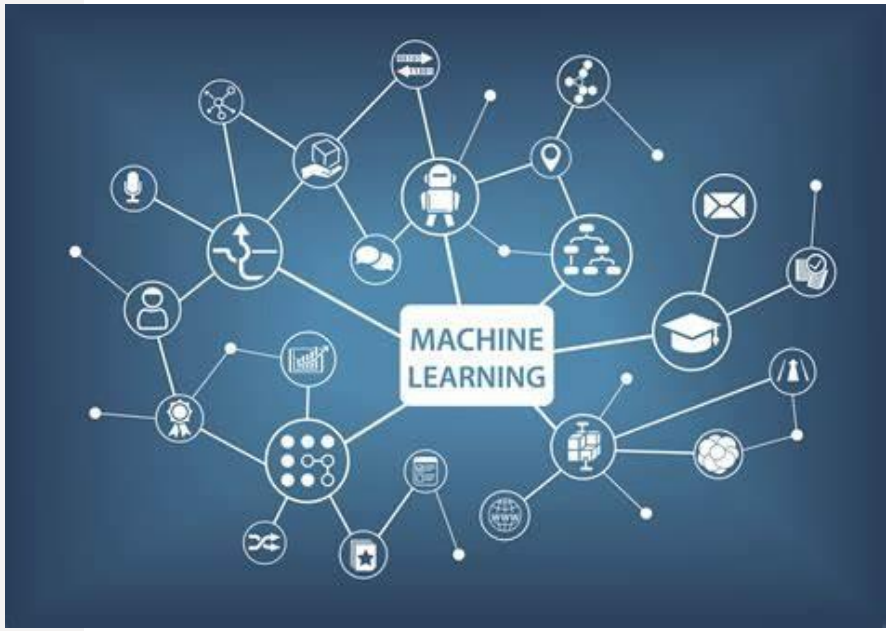
- Necesario para mantener solo variables numéricas y pueda ser válido para insertar nuestros datos categóricos-numéricos en nuestros modelos de regresión.
- Usamos el Ordinal Encoder.

	Age	Gender	Education Level	Job Title	Years of Experience	Salary	\
0	32.0	1.0	0.0	153.0	5.0	90000.0	
1	28.0	0.0	1.0	16.0	3.0	65000.0	
2	45.0	1.0	2.0	125.0	15.0	150000.0	
3	36.0	0.0	0.0	97.0	7.0	60000.0	
4	52.0	1.0	1.0	19.0	20.0	200000.0	
..	...	...	...	...	...	...	
370	35.0	0.0	0.0	126.0	8.0	85000.0	
371	43.0	1.0	1.0	27.0	19.0	170000.0	
372	29.0	0.0	0.0	66.0	2.0	40000.0	
373	34.0	1.0	0.0	132.0	7.0	90000.0	
374	44.0	0.0	2.0	105.0	15.0	150000.0	

	salario_anual	nivel de experiencia	Edad de inicio laboral	\
0	18000.000000	1.0	27.0	
1	21666.666667	1.0	25.0	
2	10000.000000	0.0	30.0	
3	8571.428571	1.0	29.0	
4	10000.000000	0.0	32.0	

PROCESAMIENTO REVISIÓN

- Nuestros datos quedaron limpios sin outliers y sin datos nulos.
- Podemos cambiar la cardinalidad con agrupación.
- Podemos hacer un posible drop con la correlación más baja en la matriz.



MODELOS DE REGRESIÓN

- REGRESIÓN LINEAL
- REDES NEURONALES
- ÁRBOLES DE DECISIÓN
- RANDOM FOREST

TRAIN /TEST y ESCALAMIENTO

- Dividiremos nuestros datos en 70-30, todos los datos con la misma división.
- Escalamos nuestros datos ya que tenemos valores que no son del mismo rango, pero no es necesario escalar en todos los modelos, pero lo dejamos estándar.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30, random_state=5)
```

```
X_train = sc.fit_transform(X_train)
X_test = sc.transform (X_test)
```

REGRESIÓN LINEAL

- No fueron necesarios los hiperparámetros, ya que regresión lineal no cuenta con ellos.

```
#inicializar el modelo
model_lr = LinearRegression()

#entrenar el modelo
model_lr.fit(X_train,y_train)

#predecir
salario_pred = model_lr.predict(X_test)

#caclular el MSE (error cuadratico medio)
mse= mean_squared_error(y_test, salario_pred) #datos reales vs los que predecimos

print('MSE', mse) #mienttas mas pequeño mejor

r2 = r2_score(y_test, salario_pred)
print("R2", r2)
```

```
MSE 219795474.49291068
R2 0.9043206319791908
```

CROSS VALIDATION REGRESIÓN LINEAL

- Hacemos cross validation para probar más particiones y usamos 10 splits.
- Mejoró pero muy poquito.

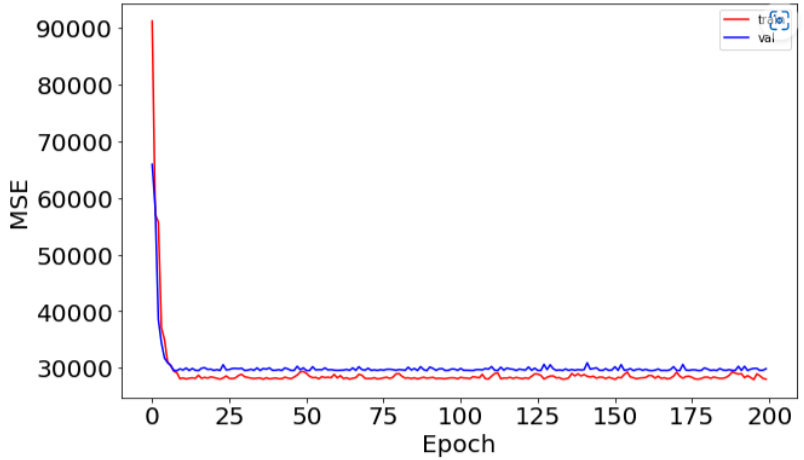
```
# Preparamos el cross validation
cv = KFold(n_splits=10,random_state=1, shuffle=True)

# crear modelo
model_lr = model_lr
# evaluar modelo
scores = -cross_val_score(model_lr, X_train, y_train, scoring='neg_mean_squared_error', cv=cv)
scores_r2 = cross_val_score(model_lr, X_train, y_train, scoring='r2', cv=cv)
# Performance
print('MSE: %.4f' % (np.mean(scores)))

print("R2", np.mean(scores_r2))
```

```
MSE: 207071145.0297
R2 0.9047355989583119
```

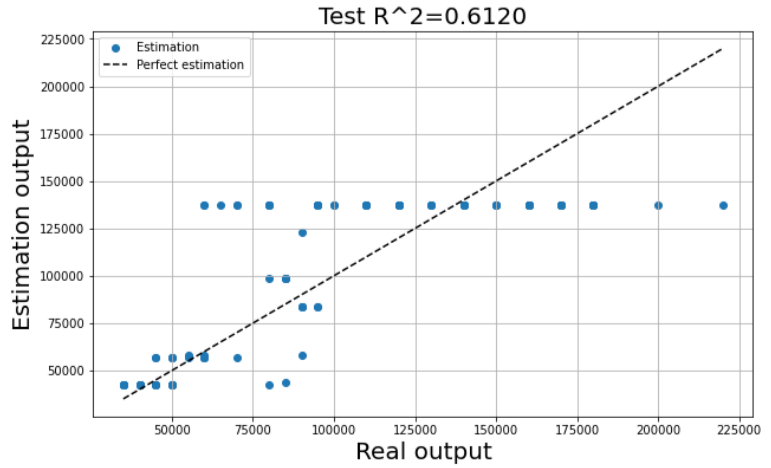
REDES NEURONALES

- Usamos la función de activación de tangente hiperbólica y la función de salida lineal para modelos de regresión.

```
#Inicializo objeto
model = Sequential()

#Agregar capas

#Agregamos la primera capa de entrada
model.add(Dense(5, activation='tanh', input_dim=9,name='input_layer')) #tanh
#Si quisiera añadir una capa oculta extra
#model.add(Dense(10, activation='tanh', name='hidden_layer'))
#Agregamos la capa de salida
model.add(Dense(1, activation='linear',name='output_layer'))

#hiperparámetros
learning_rate=0.1 #tasa de aprendizaje
epochs = 200
momentum = 0.8
decay_rate = learning_rate/epochs
sgd = SGD(lr=learning_rate, decay=decay_rate, momentum=momentum)

model.compile(loss='mean_squared_error',
              optimizer=sgd,
             metrics = ['mse'])
```




```
Test mse: 891310016.0
```




```
R2: 0.612003026260224
```

GRIDSEARCH REDES NEURONALES

- Nuestros datos fueron malos, una R2 muy baja, buscaremos el mejor Learning Rate y el mejor Moments.

```
Mejores parámetros:
{'lr': 0.05, 'momentum': 0.8}
Mejores score:
0.7558379998451149
```

```python
#Diseñar modelo con métricas optimizadas
epochs = 200
learning_rate = 0.05
decay_rate = learning_rate/epochs
momentum = 0.8

# Red neuronal
model = Sequential()
model.add(Dense(5,activation='tanh',input_dim=9))
model.add(Dense(1,activation='linear'))

# Optimizer configuration
#gradiente descendente
opt = keras.optimizers.SGD(lr=learning_rate,momentum=momentum,
                           decay=decay_rate,nesterov=True)
model.compile(loss = 'mean_squared_error',
              optimizer=opt,
              metrics=['mse'])

#Ajustar modelo
model_history = model.fit(X_train,y_train,
                    epochs=epochs,
                    batch_size=200,
                    validation_data=(X_test,y_test))

#performance
score = model.evaluate(X_test,y_test)
print('Test loss:', score[0])
print('Test mse:', score[1])
```