U N I V E R S I D A D D E P I U R A

FACULTAD DE INGENIERÍA

Análisis de datos con Python Nivel 2

Trabajo Final

Grupo: 7

Integrantes:

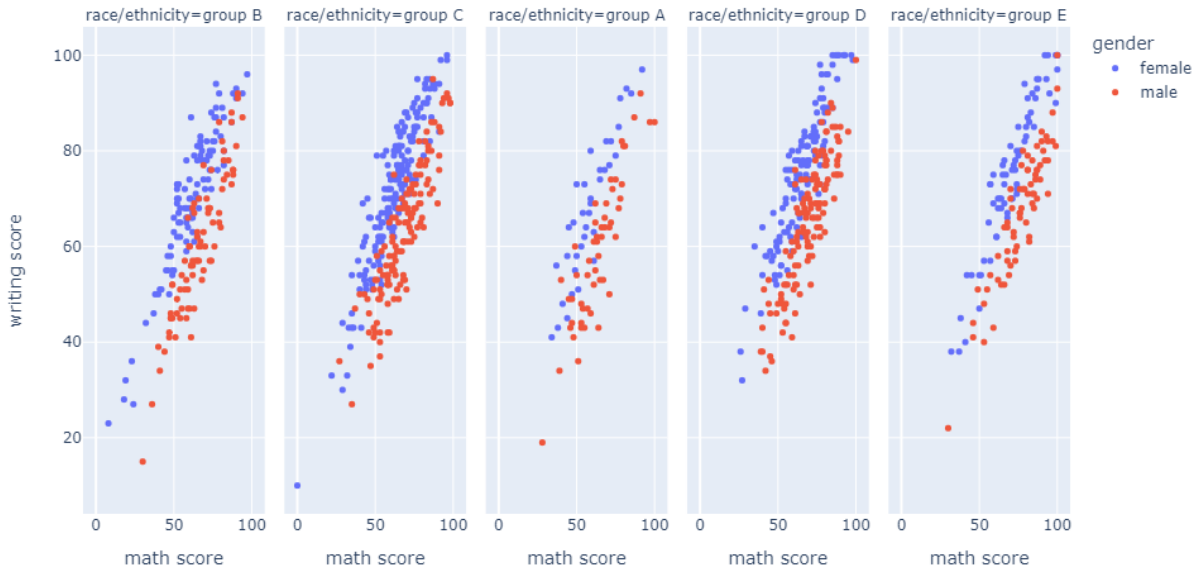
- Cobeñas Vasquez, Frank Antony
- Gambini Gamboa, Ricardo Jesús
- Ipanaqué Hau Yon, Álvaro Daniel
- Saldarriaga Valencia, Hugo Franco
- Tipe Verástegui, Christian Andrés
- Vásquez Carty, Alexander André

Introducción:

En este informe se utilizo una base de datos llamada “Students Performance in Exams” el cual tiene los datos de los resultados de los alumnos, asi como, género, raza, nivel parental de educación, si se preparo para el examen o no, entre otros. Esta información se utilizo para generar gráficos para asi poder comparar ciertas variables (se mostrarán en el informe); luego, se creo un cuatro modelos con cuatro algoritmos de Machine learning, ademas de entrenar los respectivos modelos. Se analizo los resultados de la creación y entrenamiento del modelo y también se hizo predicciones de ciertas variables en algunos modelos.

Writing Score vs Math Score:

Primero comparamos las columnas Writing Score vs Math Score teniendo en cuenta la raza o etnicidad y también el género:

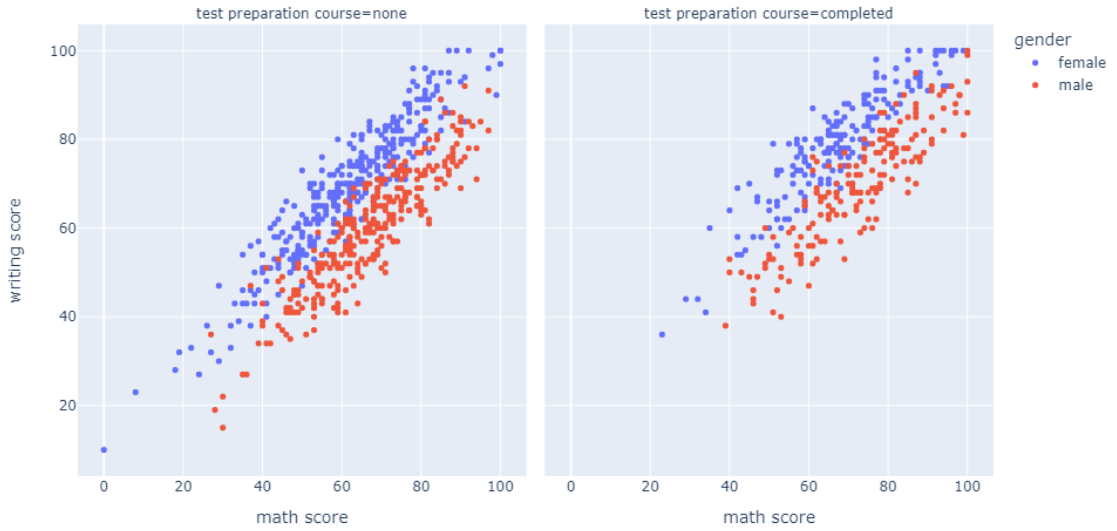


Aquí observamos como las Mujeres por termino medio llegan a obtener mejores calificaciones que los hombres en writing score; sin embargo, los hombres en promedio llegan a tener mejores calificaciones en math score que las mujeres.

Respecto a la raza y etnicidad nos podemos percatar que por término medio el Grupo E es aquel que tiene mayor Math Score y Writing Score respecto a las demás etnias o razas, sin embargo, se tiene una menor densidad de muestra respecto al Grupo C.

Por otro lado, a la raza y etnicidad nos podemos percatar que por término medio el Grupo B es aquel que tiene menor Math Score y Writing Score respecto a las demás etnias o razas.

Ahora comparamos las columnas Writing Score vs Math Score teniendo en cuenta la preparación de la persona y también el género:

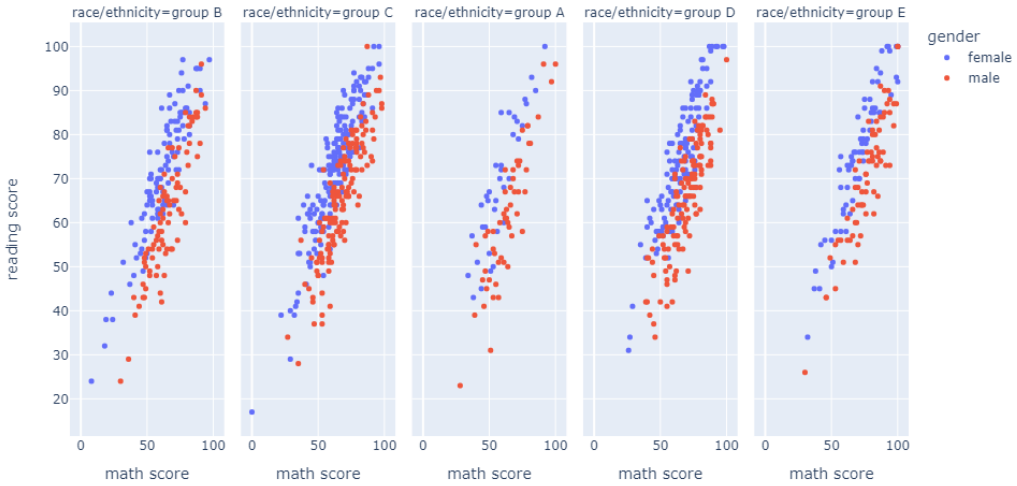


Respecto a estos gráficos de dispersión podemos darnos cuenta rápidamente que aquellos que se han preparado para el test han obtenido un mayor math score y writing score respecto a los que no lo han hecho.

Además, seguimos apreciando que las Mujeres tienen mejor Writing Score que los hombres y viceversa.

Reading Score vs Math Score

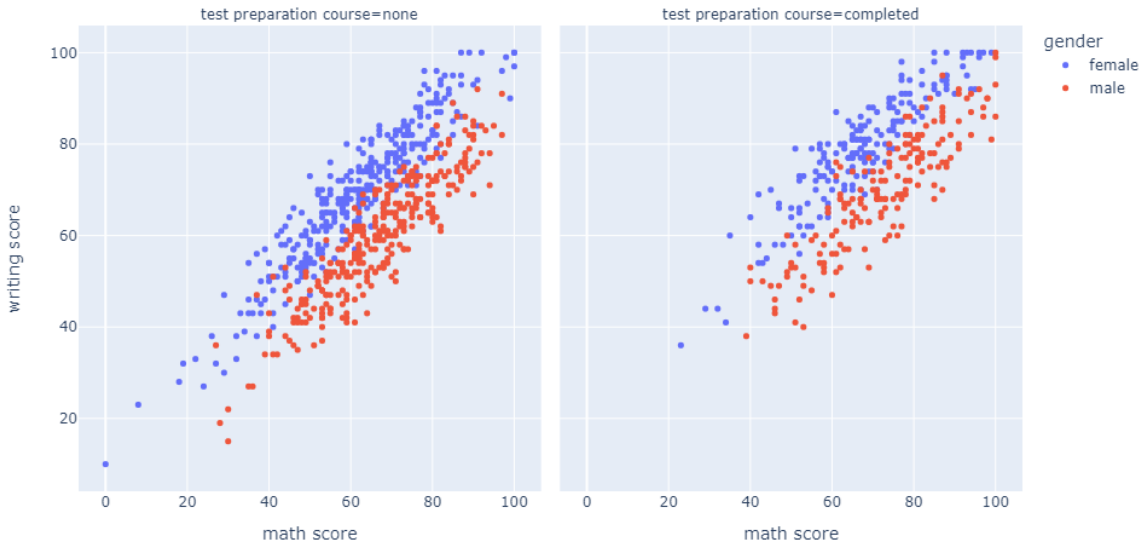
Ahora comparamos las columnas Reading Score vs Math Score teniendo en cuenta la raza o etnicidad y también el género:



De manera análoga al anterior caso, las Mujeres por término medio llegan a obtener mejores calificaciones que los hombres en Reading score; sin embargo, los hombres en promedio llegan a tener mejores calificaciones en math score que las mujeres.

Respecto a las etnias no se puede obtener una conclusión clara al respecto.

Comparamos las columnas Reading Score vs Math Score teniendo en cuenta la preparación de la persona y también el género:



Respecto a este gráfico podemos observar que hay más personas que no se han preparado y que han obtenido por término medio menores resultados en Math Score y Writing Score respecto a los que sí se han preparado.

Writing score VS Reading score:

Finalmente, comparamos las columnas Writing score vs Reading score teniendo en cuenta la raza o etnicidad y también el género:

Se puede observar que a pesar de que haya poca población de mujeres, si hablamos de término medio las mujeres presentan mejores calificaciones en Writing y Reading, siendo el grupo C y D los que más destacan en el mejor promedio de mujeres con mejores calificaciones.

Comparando las columnas Writing score vs Reading score teniendo en cuenta la preparación de la persona y también el género:

Respecto a este gráfico podemos observar que hay más personas que no se han preparado para el test, por lo tanto, en sus calificaciones en promedio presentan menores notas que las que sí se prepararon. Adicionalmente, las mujeres siguen dominando con mayores notas en término medio comparándolos con los hombres.

Analisis del problema:

- **Primer modelo – Regresión lineal:**
  Lo primero que se hizo fue importar las librerías como Pandas, Numpy, sklearn.preprocessing (importando LabelEncoder), sklearn.linear_model (importando LinearRegression) y sklearn.model_selection (importando train_test_split).

```
#Modelo 1: Regresión lineal:
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split as tts
df_students = data
# print(df_students)
```

Luego creamos una lista de valores usando np.array, para así crear una array apartir de la lista de estudiantes que presentamos anteriormente en las tablas.

```
x_gender_ = np.array(df_students['gender'])
x_race_ = np.array(df_students['race/ethnicity'])
x_course_ = np.array(df_students['test preparation course'])
x_reading = np.array(df_students['reading score'])
x_writing = np.array(df_students['writing score'])
```

Después, con el LabelEncoder() ayudamos a categorizar las etiquetas en números, ya que con el fit_transform entra en codificación y transforma el conjunto de alumnos en números.

```
label_encoder = LabelEncoder()
x_gender = label_encoder.fit_transform(x_gender_)
x_race = label_encoder.fit_transform(x_race_)
x_course = label_encoder.fit_transform(x_course_)

x = np.c_[x_gender, x_race, x_course, x_reading, x_writing]
y = np.array(df_students['math score'])
```

Continuamente, con tts nos permite dividir el datasel en bloques de entrenamineto y prueba, con el LinearRegresion () y linear_model.fit nos ayudará hacer una regresión lineal. Además, con el linear_model.predict(x) nos brinda la precisión usando el modelo lineal, y el linear_model.score(x,y) lo usaríamos para devolver el coeficiente de predición, y para que se visualicen usamos print.

```
xtrain, xtest, ytrain, ytest = tts(x, y, test_size = 0.20, random_state = 42)

linear_model = LinearRegression()
linear_model.fit(xtrain, ytrain)

y_pr_train, y_pr_test = linear_model.predict(xtrain), linear_model.predict(xtest)

score_total_train = linear_model.score(xtrain, ytrain)
score_total_validation = linear_model.score(xtest, ytest)
print("El score en entrenamiento es ", round(score_total_train, 4))
print("El score en validación es ", round(score_total_validation, 4))
```

Finalmente, para que la persona pueda ingresar datos usamos input en nuestro modelo, teniendo en cuenta cuando sería un número entero (int) o flotante (float).

```
X_genero_ = int(input('Ingrese el género (male = 1/female = 0): '))
X_raza_ = int(input('Ingrese el grupo de raza/etnia (A = 0/B = 1/C = 2/D = 3): '))
X_curso_ = int(input('Ingrese si completó curso de preparación (none = 1/completed = 0): '))
X_lectura = float(input('Ingrese el puntaje de lectura: '))
X_escritura = float(input('Ingrese el puntaje de escritura: '))

Xpre = [[X_genero_, X_raza_, X_curso_, X_lectura, X_escritura]]

prediccion = linear_model.predict(Xpre)
print('La predicción de Math Score es:', int(prediccion))
```

Este sería un ejemplo de como usar el primer modelo:

```
El score en entrenamiento es  0.8483
El score en validación es  0.8789
Ingrese el género (male = 1/female = 0): 0
Ingrese el grupo de raza/etnia (A = 0/B = 1/C = 2/D = 3): 2
Ingrese si completó curso de preparación (none = 1/completed = 0): 1
Ingrese el puntaje de lectura: 88
Ingrese el puntaje de escritura: 90
La predicción de Math Score es: 81
```

- **Segundo modelo - Ridge:**

Lo primero fue importar las librerías como Pandas, Numpy, sklearn.preprocessing (importando LabelEncoder), sklearn.linear_model (importando Ridge) y sklearn.model_selection (importando train_test_split).

```
#Modelo 2:Ridge:
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split as tts
```

Luego creamos una lista de valores usando np.array, para así crear una array apartir de la lista de estudiantes que presentamos anteriormente en las tablas.

```
x_gender_ = np.array(df_students['gender'])
x_race_ = np.array(df_students['race/ethnicity'])
x_course_ = np.array(df_students['test preparation course'])
x_reading = np.array(df_students['reading score'])
x_writing = np.array(df_students['writing score'])
```

Después, con el LabelEncoder() ayudamos a categorizar las etiquetas en números, ya que con el fit_transform entra en codificación y transforma el conjunto de alumnos en números. Adicionando que el tts, nos ayuda dividir el datasel en bloques de entrenamineto y prueba.

```
label_encoder = LabelEncoder()
x_gender = label_encoder.fit_transform(x_gender_)
x_race = label_encoder.fit_transform(x_race_)
x_course = label_encoder.fit_transform(x_course_)

x = np.c_[x_gender, x_race, x_course, x_reading, x_writing]
y = np.array(df_students['math score'])

xtrain, xtest, ytrain, ytest = tts(x, y, test_size = 0.20, random_state = 42)
```

Continuando, con el Ridge () y linear_model.fit nos ayudará hacer el metodo de Ridge regression, que en el caso de linear_model.fit nos ayuda a hacer el ajuste del modelo de regresión de crecimiento. Además, con el linear_model.predict(x) nos brinda la precisión usando el modelo lineal, y el linear_model.score(x,y) lo usaríamos para devolver el coeficiente de predición, y para que se visualicen usamos print.

```
linear_model = Ridge()
linear_model.fit(xtrain, ytrain)

y_pr_train, y_pr_test = linear_model.predict(xtrain), linear_model.predict(xtest)


score_total_train = linear_model.score(xtrain, ytrain)
score_total_validation = linear_model.score(xtest, ytest)
print("El score en entrenamiento es ", round(score_total_train, 4))
print("El score en validación es ", round(score_total_validation, 4))
```

Finalmente, para que la persona pueda ingresar datos usamos input en nuestro modelo, teniendo en cuenta cuando sería un número entero (int) o flotante (float).

```
X_genero_ = int(input('Ingrese el género (male = 1/female = 0): '))
X_raza_ = int(input('Ingrese el grupo de raza/etnia (A = 0/B = 1/C = 2/D = 3): '))
X_curso_ = int(input('Ingrese si completó curso de preparación (none = 1/completed = 0): '))
X_lectura = float(input('Ingrese el puntaje de lectura: '))
X_escritura = float(input('Ingrese el puntaje de escritura: '))

Xpre = [[X_genero_, X_raza_, X_curso_, X_lectura, X_escritura]]

prediccion = linear_model.predict(Xpre)
print('La predicción de Math Score es:', int(prediccion))
```

De esta manera, se les presenta el resultado de la ejecución del segundo modelo:

```
El score en entrenamiento es  0.8483230531179223
El score en validación es  0.8788559576676997
```

- **Tercer modelo:**

Primero se importó las librerías como Pandas, Numpy, sklearn.preprocessing (importando LabelEncoder), sklearn.ensemble (importando RandomForestRegressor) y sklearn.model_selection (importando train_test_split).

```
#Modelo 3: Random Forest Regressor:
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split as tts
```

Después, creamos una lista de valores usando np.array, para así crear una array apartir de la lista de estudiantes que presentamos anteriormente en las tablas.

```
x_gender_ = np.array(df_students['gender'])
x_race_ = np.array(df_students['race/ethnicity'])
x_course_ = np.array(df_students['test preparation course'])
x_reading = np.array(df_students['reading score'])
x_writing = np.array(df_students['writing score'])
```

Después, con el LabelEncoder() ayudamos a categorizar las etiquetas en números, ya que con el fit_transform entra en codificación y transforma el conjunto de alumnos en números. Adicionando que el tts, nos ayuda dividir el datasel en bloques de entrenamineto y prueba.

```
label_encoder = LabelEncoder()
x_gender = label_encoder.fit_transform(x_gender_)
x_race = label_encoder.fit_transform(x_race_)
x_course = label_encoder.fit_transform(x_course_)

x = np.c_[x_gender, x_race, x_course, x_reading, x_writing]
y = np.array(df_students['math score'])

xtrain, xtest, ytrain, ytest = tts(x, y, test_size = 0.20, random_state = 42)
```

Continuando el código, con RandomForestRegressor () y a la par con RFR_model.fit(x,y) se usaron para armar un “bosque de arboles” como su nombre lo dice para poder construir un conjunto de entrenamiento. Agregando, que se usaría el .predict() para predecir la regresión, y el .score() para devolver el coeficiente de predición, para luego imprimir con print el score en entrenamiento y validación.

```
RFR_model = RandomForestRegressor(random_state=42)
RFR_model.fit(xtrain, ytrain)

y_pr_train, y_pr_test = RFR_model.predict(xtrain), RFR_model.predict(xtest)

score_total_train = RFR_model.score(xtrain, ytrain)
score_total_validation = RFR_model.score(xtest, ytest)
print("El score en entrenamiento es ", round(score_total_train, 4))
print("El score en validación es ", round(score_total_validation, 4))
```

Finalmente, para poder ingresar datos usamos input en nuestro modelo, teniendo en cuenta cuando sería un número entero (int) o flotante (float), y print para imprimir la predición de Math score.

```
X_genero_ = int(input('Ingrese el género (male = 1/female = 0): '))
X_raza_ = int(input('Ingrese el grupo de raza/etnia (A = 0/B = 1/C = 2/D = 3): '))
X_curso_ = int(input('Ingrese si completó curso de preparación (none = 1/completed = 0): '))
X_lectura = float(input('Ingrese el puntaje de lectura: '))
X_escritura = float(input('Ingrese el puntaje de escritura: '))

Xpre = [[X_genero_, X_raza_, X_curso_, X_lectura, X_escritura]]

prediccion = linear_model.predict(Xpre)
print('La predicción de Math Score es:', int(prediccion))
```

Un ejemplo de cómo se vería nuestro tercer modelo en uso sería:

```
El score en entrenamiento es  0.9726
El score en validación es  0.8396
Ingrese el género (male = 1/female = 0): 1
Ingrese el grupo de raza/etnia (A = 0/B = 1/C = 2/D = 3): 2
Ingrese si completó curso de preparación (none = 1/completed = 0): 0
Ingrese el puntaje de lectura: 88
Ingrese el puntaje de escritura: 90
La predicción de Math Score es: 91
```

- **Cuarto modelo:**

Para el último código, se repite el mismo proceso que los 3 códigos anteriores, que serpia importar, pero en este caso lo nuevo sería de la librería sklearn.ensemble importamos GradientBoostingRegressor, que sería un conjuto de armboles de decisión. Adicionando que en la línea 24, tambíen habría un cambio que para que el modelo funcione seríaRFR_model = GradientBoostingRegressor(x) para así poder usar la librería que hemos importado.

```python
#Modelo 4: GradientBoostinRegressor:
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split as tts
from sklearn.ensemble import GradientBoostingRegressor

x_gender_ = np.array(df_students['gender'])
x_race_ = np.array(df_students['race/ethnicity'])
x_course_ = np.array(df_students['test preparation course'])
x_reading = np.array(df_students['reading score'])
x_writing = np.array(df_students['writing score'])

label_encoder = LabelEncoder()
x_gender = label_encoder.fit_transform(x_gender_)
x_race = label_encoder.fit_transform(x_race_)
x_course = label_encoder.fit_transform(x_course_)

x = np.c_[x_gender, x_race, x_course, x_reading, x_writing]
y = np.array(df_students['math score'])

xtrain, xtest, ytrain, ytest = tts(x, y, test_size = 0.20, random_state = 42)

RFR_model = GradientBoostingRegressor(random_state=42, max_depth=4)
RFR_model.fit(xtrain, ytrain)

y_pr_train, y_pr_test = RFR_model.predict(xtrain), RFR_model.predict(xtest)

score_total_train = RFR_model.score(xtrain, ytrain)
score_total_validation = RFR_model.score(xtest, ytest)
print("El score en entrenamiento es ", round(score_total_train, 4))
print("El score en validación es ", round(score_total_validation, 4))

X_genero_ = int(input('Ingrese el género (male = 1/female = 0): '))
X_raza_ = int(input('Ingrese el grupo de raza/etnia (A = 0/B = 1/C = 2/D = 3): '))
X_curso_ = int(input('Ingrese si completó curso de preparación (none = 1/completed = 0): '))
X_lectura = float(input('Ingrese el puntaje de lectura: '))
X_escritura = float(input('Ingrese el puntaje de escritura: '))

Xpre = [[X_genero_, X_raza_, X_curso_, X_lectura, X_escritura]]

prediccion = linear_model.predict(Xpre)
print('La predicción de Math Score es:', int(prediccion))
```

Finalmente ejecutando el código, nos saría el siguiente resultado:

```
El score en entrenamiento es  0.9146408569940696
El score en validación es  0.871899046468426
```

Conclusiones:

- Gracias a este curso se ha podido aprender a solucionar problemáticas reales mediante el uso del lenguaje de programación Python y así poder automatizar ciertos procesos que muchas veces realizamos de manera manual y secuencial.

- Se puede deducir que respecto a los gráficos de las calificaciones en Writing, Reading y Math score, las mujeres son mejores en Writing y Reading que los hombres, por otro lado, los hombres presentan mejores notas en promedio en Math score.

- Poder saber cuándo un modelo no nos es útil es importante, es decir; que el modelo no generalice a nuevos datos (sub-ajuste) o que modele los datos demasiado bien (sobreajuste). Para evitar este problema tenemos que buscar un modelo que este en un punto óptimo entre lo ya antes mencionado. No siempre es fácil poder lograr esto, debido a esto se enseño algunas formas de llegar al punto adecuado, esto puede ser: entrenar la data, etc.

- El previo conocimiento de programación básica es importante, debido a que gracias a esto se puede entender mejor los códigos que se usaran para machine learning; también para poder corregir errores comunes o que hayamos podido cometer durante la creación de un modelo, etc.

- Es importante no tener muchos datos atípicos sino más bien contar con una considerable cantidad de datos relevantes, que contribuyan adecuadamente al modelo de machine learning y para que, de esta forma, se realice un buen entrenamiento de la información.