Tipología y ciclo de vida del dato.

Práctica 2: Limpeza de datos

Raúl Cacho Martínez

Máster en Data Science

Universitat Oberta de Catalunya

1 de enero de 2019

Índice

1. Introducción

Este documento, junto con el fichero practica2.py de Python (versión 3.6.6), el fichero practica.ipynb de Jupyter Notebook, el fichero practica2.pdf resultado de la ejecución del Notebook y el fichero datos_F1_1950_2018_prac2.csv pueden encontrarse en el siguiente enlace de Github:

`https://github.com/raulcacho/DataCleaning/`

Esta práctica consiste en la limpieza de datos obtenidos en la Práctica 1. Estos datos consistían en el histórico de resultados de las carreras de Fórmula 1 disputadas desde 1950 hasta la actualidad.

En esta práctica se tratará de limpiar los datos obtenidos haciéndolos utilizables en el proceso de análisis, ya que se han detectado algunos problemas que no permiten usarlos tal y como se extrajeron de la web.

El planteamiento de la práctica es, en primer lugar, identificar cuales son los problemas detectados, indicando su origen y como pueden afectar al posterior análisis.

Una vez detectados, se describirá como solucionarlos. En caso de no poder solucionarlos, se indicará qué decisión se toma respecto al dato, en función del impacto que pueda tener en el análisis.

2. Descripción del conjunto de datos

El conjunto de datos recoge los resultados de todas las carreras (denominadas Gran Premio, GP) de la historia de la Fórmula 1 desde sus orígenes en 1950 hasta la última temporada disputada (2018). Este conjunto de datos es el resultado de la Práctica 1 de esta asignatura, y puede descargarse en `https://github.com/raulcacho/WebScraping/blob/master/datos_F1_1950_2018.csv`.

Es un fichero .csv que emplea el caracter "," como separador. En total consta de 23009 filas, de las cuales la primera es la cabecera. Cada una de las filas consta de 10 campos:

Campo	Descripción
Year	Año en el que se disputó el Gran Premio
GP	Nombre del Gran Premio
Driver	Piloto (Pilotos) que pilotaron el coche
Number	Dorsal con el que el piloto disputó el Gran Premio
Team	Escudería con la que compitió el piloto
Grid position	Posición en la que inició la carrera
Final position	Posición en la que finalizó la carrera
Points	Puntos obtenidos en la carrera
Finish	Tiempo empleado en acabar la carrera/motivo por el que no se finalizó
Laps	Vueltas dadas durante la carrera

2.1. Problemas detectados

En un análisis preliminar del conjunto de datos se han encontrado los siguientes problemas:

1. Algunas filas tienen diferente número de columnas: Esto se debe a que el campo “Finish”, incluye el caracter “,” que es el mismo que se ha empleado como separador de campos. Esto sucede, por ejemplo, en la línea 757", donde encontramos lo siguiente: “1954, argentina, Jean Behra,18,Gordini,17,10,0, Disqualified, Push Start, 61”.

2. El último campo de algunas filas está repetido varias veces. Por ejemplo, en la línea 11 aparece “3232”. Esto claramente es un error, en comparación con el número de vueltas dado por el resto de pilotos. Un análisis cuidadoso de estos datos, revela que estos valores extremos se deben a que hay varios pilotos en la misma línea, y el número de vueltas aparece una vez por cada piloto. En la línea 11 hay 2 pilotos, mientras que, por ejemplo, en la línea 777 hay 5 pilotos y aparece el valor “200200200200200”.

3. Relacionado con el punto anterior, hay pilotos de varias nacionalidades. Esto tiene como consecuencia que hay pilotos cuyo nombre se escribe con un nombre y un apellido (por ejemplo: Nino Farina) y otros con nombre y dos apellidos (Hernando da Silva Ramos). Algunos pilotos tienen nombre compuesto (Juan Manuel Fangio) y otros tienen apellidos con preposiciones (Wolfgang von Trips). Esto dificulta la división del campo de pilotos para poder obtener información de qué pilotos compartieron coche en cada carrera.

4. En el campo “GP”, se indica el nombre del Gran Premio disputado. En general, toman el nombre del país o la ciudad en que se ubica el circuito. Sin embargo, algunos países reciben denominaciones diferentes. Por ejemplo, Estados Unidos aparece como “the united states” y “us”.

5. Problemas de codificación. Se ha visto que hay algunos caracteres no están soportados, en particular, los caracteres acentuados.

6. Algunos campos están vacíos. La mayor parte de las ocasiones se trata del campo relacionado con la posición de inicio. Generalmente se trata de eventos en los que el piloto no tomó la salida. El que este campo esté vacío, además indica que el siguiente no es fiable. Por la forma en la que está construido el dataset, el campo “Final position” siempre tendrá un valor, aunque no debería ser así si el piloto no ha tomado la salida.

2.2. Pregunta a responder

Con este conjunto de datos, intentaremos obtener una gráfica que muestre qué pilotos han obtenido más puntos en la historia de la competición. También podremos ver si existe correlación entre la posición de salida y final en una carrera.

3. Limpieza de los datos

3.1. Lectura del fichero de datos

En primer lugar, procederemos a leer los datos del fichero[1]. En este paso convertiremos los datos en un data frame, para que sea más cómodo trabajar con ellos. Para ello, tenemos que solucionar el problema de que hay filas con más campos, debido a que el campo “Finish” puede tener el caracter “,”, y al leer el fichero se interpreta como dos campos independientes.

```
# Abrimos el fichero con los datos
file = open('datos_F1_1950_2018_prac2.csv', 'r', encoding="utf-16")

# Leemos el fichero
data = file.readlines()

#>Cerramos el fichero
file.close()
```

[1]Esta práctica se realizará usando Python 3.7. En el directorio de github se acompaña el código en python así como un fichero de Jupyter notebook

```
# Eliminamos los caracteres de saltos de línea y dividimos por ","
data = [(d.strip('\n')).split(',') for d in data]

# Corregimos aquellas filas en las que hay una "," en el campo Finish
for d in data:
    if len(d) > 10:
        d[8] = d[8] + ':' + d[9]
        d.pop(9)

# Convertimos los datos en Data Frame
df_data = pd.DataFrame.from_records(data[1:], columns=data[0])

# Lo mostramos en pantalla
df_data
```

Al mostrar los datos en pantalla, vemos que hay caracteres que se han codificado de forma incorrecta. Para eliminarlos y sustituirlos por los caracteres adecuados, ejecutamos el siguiente código:

```
# Eliminamos los caracteres erróneos
for j in range(len(df_data)):
    dummy = df_data['Driver'][j]
    df_data['Driver'][j] = df_data['Driver'][j].replace(''Ã‰'', 'É')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã¨', 'è')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã©', 'é')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã¡', 'á')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã ', 'á')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã“', 'Ó')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã³', 'ó')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã¤', 'ä')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã¶', 'ö')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã¼', 'ü')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã', 'í')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã§', 'ç')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã´', 'ô')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã¸', 'ø')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ãº', 'ú')
    df_data['Driver'][j] = df_data['Driver'][j].replace('Ã±', 'ñ')
```

3.2. Corrección de registros

Una vez tenemos los caracteres correctos, nos encontramos con el problema de que en el campo "Driver" podemos encontrar uno o más pilotos, que no están separados por ningún carácter particular. Además, cada piloto puede tener nombre simple o compuesto y uno o dos apellidos, lo cual dificulta la tarea de separarlos. Para poder hacerlo, seguimos el siguiente proceso:

3.2.1. Estimación de la cantidad de nombres en campo "Driver"

En primer lugar, estimamos la cantidad de nombres que hay en cada campo. En este paso supondremos que el nombre del piloto consta, al menos, de nombre y apellido. Para estimar el número de nombres, separamos la cadena de texto según los espacios, dividimos entre dos y aproximamos al entero más bajo. De aquí, podemos sacar una primera lista de todos los pilotos que han competido en Fórmula 1. El código empleado es el siguiente:

```
# Estimamos el número de pilotos en cada campo
N_drivers = [math.floor(len(d.split())/2) for d in df_data['Driver']]

# Buscamos aquellos campos con un solo nombre
indice_1  = np.array(N_drivers)==1
drivers_list = list(set(df_data['Driver'][indice_1]))
```

3.2.2. Lista preliminar de pilotos

La lista anterior sólo contiene aquellos pilotos que han competido, al menos una vez, en solitario, y cuyo nombre consta de nombre y apellido, nombre compuesto y un apellido o un nombre y dos apellidos. Se quedan fuera de esta lista aquellos pilotos que siempre han competido en equipo, o cuyo nombre completo consta de más de tres palabras (por ejemplo, *Pedro de la Rosa*). Con esta primera lista, podremos hacer una primera iteración para separar los pilotos dentro del mismo campo, ejecutando el siguiente código:

```
# Creamos una lista donde almacenar el resultado
sep_drivers = []

# Para cada registro...
for i in range(len(df_data['Driver'])):
    # Definimos una variable para simplificar el código
    d = df_data['Driver'][i]
```

```
# Variable auxiliar donde almacenamos la posición
# inicial del nombre de los pilotos
limits = []

# Para cada nombre en la lista...
for dr in drivers_list:
    # Comprobamos si el nombre está en el registro
    if dr in d:
        # En caso afirmativo, lo almacenamos
        limits.append(d.find(dr))

# Ordenamos las posiciones de forma ascendente
limits = sorted(limits)
# Y añadimos la longitud de la cadena
limits.append(len(d))

# Creamos una variable para almacenar los resultados parciales
dummy = []
# Extraemos los nombres del campo del registro
for j in range(len(limits)-1):
    dummy.append(d[limits[j]:limits[j+1]].strip(' '))

# Limpiamos los resultados
dummy = [st for st in dummy if st != '']

# Y los añadimos a la variable creada para ello
sep_drivers.append(dummy)
```

3.2.3. Detección de pilotos no incluidos en la lista

Sin embargo, esta separación contiene errores, debido a que no todos los nombres de los pilotos estaban incluidos en la lista obtenida en el punto 1. Estos nombres que no se han incluido, hay que introducirlos a mano en la lista. Para saber cuales son, ejecutamos el siguiente código:

```
# Creamos una variable para almacenar los pilotos que faltan en la lista
missing = []

# Para cada registro...
for i in range(len(df_data['Driver'])):
```

```
    # Comparamos la longitud del campo original con la
    # suma de las longitudes de los nombres separados
    if len(' '.join(sep_drivers[i])) != len(df_data['Driver'][i]):
        # Si son diferentes, añadimos el registro a la variable
        missing.append(df_data['Driver'][i])

# Mostramos en pantalla los pilotos que faltan
print(list(set(missing)))
```

3.2.4. Adición de pilotos faltantes en la lista

El siguiente paso es incluir estos nombres en la lista de pilotos. Para ello, ejecutamos lo siguiente:

```
drivers_list.append('Joie Chitwood')
drivers_list.append('Dries van der Lof')
drivers_list.append('Bill Cantrell')
drivers_list.append('Bayliss Levrett')
drivers_list.append('Hernando da Silva Ramos')
drivers_list.append('Carel Godin de Beaufort')
drivers_list.append('Maria Teresa de Filippis')
drivers_list.append('Mario de Araujo Cabral')
drivers_list.append('Syd van der Vyver')
drivers_list.append('Pedro de la Rosa')
drivers_list.append('Giedo van der Garde')
drivers_list.append('Eric van de Poele')
drivers_list.append('Joe Fry')
drivers_list.append('Brian Shawe Taylor')
drivers_list.append('Dorino Serafini')
drivers_list.append('Alberto Ascari')
drivers_list.append('Johnny Mantz')
drivers_list.append('Walt Faulkner')
drivers_list.append('Frank Armi')
drivers_list.append('George Fonder')
drivers_list.append('Danny Kladis')
drivers_list.append('Spider Webb')
drivers_list.append('Len Duncan')
drivers_list.append('George Fonder')
```

3.2.5. iteración del proceso

Volvemos a ejecutar el código del punto 3.2.2. Verificamos, con el código del punto 3.2.3, que no falta ningún piloto por incluir. En caso de que haga falta incluir algún nombre, repetimos 3.2.4, 3.2.2 y 3.2.3, haciendo las modificaciones oportunas hasta que estén todos los nombres incluidos. Una vez hemos comprobado que no faltan nombres por incluir, trasladamos los cambios al data frame:

```
df_data['Drivers'] = sep_drivers
```

3.3. Campos "Points", "Grid position" y "Laps"

Con estos campos sucede algo similar a lo que pasaba con el campo "Driver". En estos campos se encuentra la información de todos los pilotos incluidos en el campo "Driver" de cada registro. Para dividirlos, empleamos el siguiente código:

```
# Identificamos el número de pilotos en cada registro
n_items = [len(d) for d in df_data['Driver']]

# Creamos unas variables que almacenan los datos que nos interesan
Laps = []
Grid = []
Points = []

# Para cada registro
for i in range(len(n_items)):
    # Atributo "Laps"
    # Determinamos la longitud del atributo para cada piloto
    n=int(len(df_data['Laps'][i])/n_items[i])
    # Dividimos la cadena de texto en subcadenas de la longitud calculada
    dummy_1 = [df_data['Laps'][i][j * n:(j + 1) * n] for j in range((len(df_data[
    # Almacenamos el resultado
    Laps.append(dummy_1)

    # Atributo "Grid Position"
    # Determinamos la longitud del atributo para cada piloto
    n=int(len(df_data['Grid position'][i])/n_items[i])
    # Si el atributo está vacío
    if (len(df_data['Grid position'][i]) == 0):
        # Lo cambiamos por la palabra "Pitlane"
        dummy_2 = ['Pitlane'] * n_items[i]
    # Si el atributo tiene una longitud menor que el número de pilotos
```

```
    elif (len(df_data['Grid position'][i]) < n_items[i]):
        # Suponemos que el atributo aplica a todos ellos
        dummy_2 = [df_data['Grid position'][i]] * n_items[i]
    # En otro caso
    else:
        # Dividimos la cadena de texto en subcadenas de la longitud calculada
        dummy_2 = [df_data['Grid position'][i][j * n:(j + 1) * n] for j in range(
    # Almacenamos el resultado
    Grid.append(dummy_2)

    # Atributo "Points"
    # Determinamos la longitud del atributo para cada piloto
    n=int(len(df_data['Points'][i])/n_items[i])
    # Si el atributo está vacío
    if (df_data['Points'][i] == ''):
        # Lo cambiamos por el valor "0"
        dummy_3 = [0] * n_items[i]
    # Si el atributo tiene una longitud menor que el número de pilotos
    elif (len(df_data['Points'][i]) < n_items[i]):
        # Suponemos que el atributo aplica a todos ellos
        dummy_3 = df_data['Points'][i] * n_items[i]
    # En otro caso
    else:
        # Dividimos la cadena de texto en subcadenas de la longitud calculada
        dummy_3 = [df_data['Points'][i][j * n:(j + 1) * n] for j in range((len(df
    # Almacenamos el resultado
    Points.append(dummy_3)

# Sustituimos las columnas en el data frame por las nuevas columnas
df_data['Laps'] = Laps
df_data['Grid position'] = Grid
df_data['Points'] = Points
```

3.4. Homogeneización del campo “GP”

El campo “GP” contiene el nombre del Gran Premio. Sin embargo, el formato en el que está escrito, no es homogéneo. Por ejemplo, Estados Unidos aparece en la lista de varias formas: *us, united states, usa, the united states, ...* Aunque para los objetivos marcados esto no es relevante, se ha decidido homogeneizar los datos de este campo. Para ello se ha elegido como criterio los atributos sean países (salvo algunas carreras particulares, como Indianápolis o Mónoca) y que las palabras empiecen por todas por

mayúscula. El código es el siguieente:

```
df_data['GP'] = [d.replace('indianapolis 500', 'Indianapolis 500') for d in df_da
df_data['GP'] = [d.replace('australian', 'australia') for d in df_data['GP']]
df_data['GP'] = [d.replace('austrian', 'austria') for d in df_data['GP']]
df_data['GP'] = [d.replace('belgian', 'belgium') for d in df_data['GP']]
df_data['GP'] = [d.replace('brazilian', 'brazil') for d in df_data['GP']]
df_data['GP'] = [d.replace('british', 'great britain') for d in df_data['GP']]
df_data['GP'] = [d.replace('canadian', 'canada') for d in df_data['GP']]
df_data['GP'] = [d.replace('caesars palace', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace('chinese', 'china') for d in df_data['GP']]
df_data['GP'] = [d.replace('dallas', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace('detroit', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace('european', 'europe') for d in df_data['GP']]
df_data['GP'] = [d.replace('french', 'france') for d in df_data['GP']]
df_data['GP'] = [d.replace('german f1', 'germany') for d in df_data['GP']]
df_data['GP'] = [d.replace('germanyy', 'german') for d in df_data['GP']]
df_data['GP'] = [d.replace('hungarian', 'hungary') for d in df_data['GP']]
df_data['GP'] = [d.replace('indian', 'india') for d in df_data['GP']]
df_data['GP'] = [d.replace('italian', 'italy') for d in df_data['GP']]
df_data['GP'] = [d.replace('japanese', 'japan') for d in df_data['GP']]
df_data['GP'] = [d.replace('the pacific', 'japan') for d in df_data['GP']]
df_data['GP'] = [d.replace('korean', 'korea') for d in df_data['GP']]
df_data['GP'] = [d.replace('malaysian', 'malaysia') for d in df_data['GP']]
df_data['GP'] = [d.replace('mexican', 'mexico') for d in df_data['GP']]
df_data['GP'] = [d.replace('pescara', 'italy') for d in df_data['GP']]
df_data['GP'] = [d.replace('russian', 'russia') for d in df_data['GP']]
df_data['GP'] = [d.replace('spanish', 'spain') for d in df_data['GP']]
df_data['GP'] = [d.replace('the netherland', 'the netherlands') for d in df_data[
df_data['GP'] = [d.replace('the netherlandss', 'the netherlands') for d in df_dat
df_data['GP'] = [d.replace('the united states west', 'u.s.a.') for d in df_data['
df_data['GP'] = [d.replace('the united states', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace('united states', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace('us f1', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace('usa f1', 'u.s.a.') for d in df_data['GP']]
df_data['GP'] = [d.replace(' gp', '') for d in df_data['GP']]
df_data['GP'] = [d.replace(' f1', '') for d in df_data['GP']]

for i in range(len(df_data)):
    df_data['GP'][i] = df_data['GP'][i].title()
```

3.5. Separación de registros

Hasta ahora, hay algunos registros que incluyen la información de varios pilotos. Esto puede deberse a que son carreras por equipos, o a que hay errores en la codificación de la web de la que se extrajeron los datos, y los registros estaban en una sola fila. Sea como fuere, para el propósito de esta práctica, no es necesario en cuenta cuál es el motivo de esta circunstancia, sino que es más interesante poder dividir esos registros en tantos registros como pilotos incluya el campo "Driver". El código empleado para este fin es el siguiente:

```
# Creamos variables para almacenar los atributos de cada registro
Year   = []
GP     = []
Driver = []
Number = []
Team   = []
Grid   = []
Final  = []
Points = []
Finish = []
Laps   = []

# Para cada registro...
for i in range(len(df_data)):
    # Y por cada nombre en el campo "Driver"...
    for j in range(len(df_data['Driver'][i])):
        # Añadimos los atributos de cada registro
        # a la variable corresondiente
        Year.append(int(df_data['# Year'][i]))
        GP.append(df_data['GP'][i])
        Driver.append(str(np.array(df_data['Driver'][i][j]).flatten()[0]))
        Number.append(int(df_data['Number'][i]))
        Team.append(df_data['Team'][i])
        Grid.append(float(np.array(df_data['Grid position'][i][j]).flatten()[0]))
        Final.append(int(np.array(df_data['Final position'][i]).flatten()[0]))
        Points.append(float(np.array(df_data['Points'][i][j]).flatten()[0]))
        Finish.append(df_data['Finish'][i])
        Laps.append(int(np.array(df_data['Laps'][i][j]).flatten()[0]))

# Creamos un data frame con estas variables
df = pd.DataFrame.from_items([('Year', Year),
```

```
('GP', GP),
('Driver', Driver),
('Number', Number),
('Team', Team),
('Grid', Grid),
('Final', Final),
('Points', Points),
('Finish', Finish),
('Laps', Laps),
 ])
```

Con esto tenemos el data frame con el que proceder al análisis

3.6. Selección de los datos de interés

3.6.1. Puntuación de los pilotos

Para el análisis de los pilotos con más puntos de la historia de la competición, seleccionaremos aquellos registros sin "NA" y en los que el campo "Points" sea mayor que 0. El código para ello es:

```
selection_1 = (df['Points'] >0) & (np.isfinite(df['Points']))
```

3.6.2. Posiciones de salida y llegada

Para el modelo lineal entre la posición de salida y de llegada en una carrera, nos quedaremos con los datos de aquellos pilotos que hayan llegado a meta. Para poder discriminar, tenemos que hacer una pequeña transformación de los datos. Los registros de aquellos pilotos que terminan la carrera incluyen el tiempo que emplearon en acabarla (en formato hh:mm:ss.sss) o las vueltas respecto al ganador (+x laps). De aquellos que no acabaron, se indica el motivo. Por ello, para diferenciarlos miraremos si el primer carácter es un dígito o no. Además, puesto que compararemos dats conla posición de salida, nos interesa que no haya valores no válidos en este último campo:

```
# Suprimimos los signos '+' del format ''+x laps'
selection_2 = [d.replace('+', '') for d in df['Finish']]
# En algunos campos aparece un signo '-', también lo quitamos
selection_2 = [d.replace('-', '') for d in selection_2]
# Miramos si el primer carácter es un dígito
selection_2 = [d[0].isdigit() for d in selection_2]
# Seleccionamos también aquellos datos no nulos en la posición de salida
selection_2 = selection_2 & (np.isfinite(df['Grid']))
```

4. Análisis

4.1. Pilotos con mayor puntuación

La mejor manera de representar los pilotos con mayor puntuación acumulada es con un gráfico de barras horizontales. Una vez seleccionados los datos (ver sección 3.6.1), sumamos, para cada piloto, las puntuaciones obtenidas:

```
# creamos una lista en la que cada piloto aparezca una vez
scoring_drivers = np.array(list(set(df['Driver'][selection_1])))

# creamos una variable para almacenar los datos
drivers_score = []
# Para cada piloto...
for driver in scoring_drivers:
    # sumamos sus puntos
    dummy = np.sum(df['Points'][df['Driver']==driver])
    # y los almacenamos
    drivers_score.append(dummy)
```

Una vez sumadas las puntuaciones, nos quedamos con las 25 más altas y las representamos en un gráfico de barras:

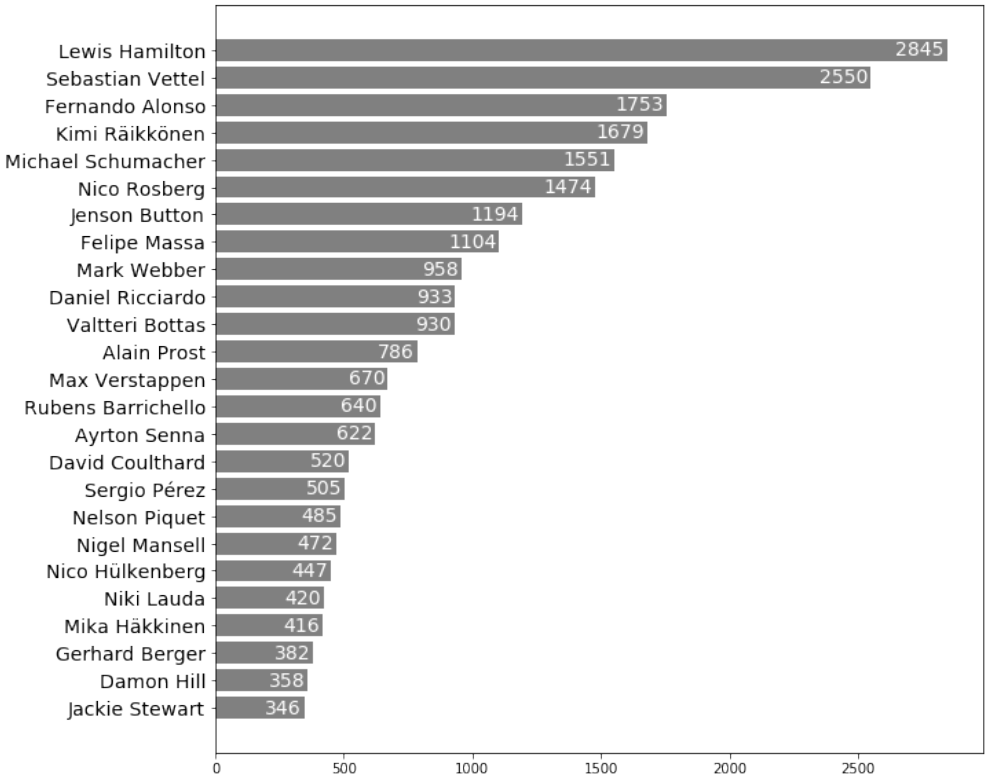

Figura 1: Pilotos con las 25 mayores puntuaciones de la historia de F1.

Podemos ver que Lewis Hamilton es el piloto que más puntos ha sumado, con 2845. Al buscar este dato, se puede ver fácilmente que Lewis Hamilton lleva sumados 3018 puntos. El origen de la diferencia no se encuentra en el proceso de limpieza de los datos, sino que se debe a errores en la web de la que se extrajeron. Por ejemplo, en la temporada del año 2000 (ver `https://www.f1-fansite.com/f1-results/2000-f1-championship-standings/`), el gran premio de los Estados Unidos, en realidad está enlazado al gran premio de Italia. Este error se ha detectado varias veces más, lo que hace que los resultados obtenidos sean diferentes a los encontrados en la literatura.

4.2. Comparación posiciones inicial y final

La selección de estos datos puede verse en la sección 3.6.2. En este caso optaremos por un histograma bidimensional. Realizaremos un ajuste lineal para ver si existe correlación entre ambas variables.

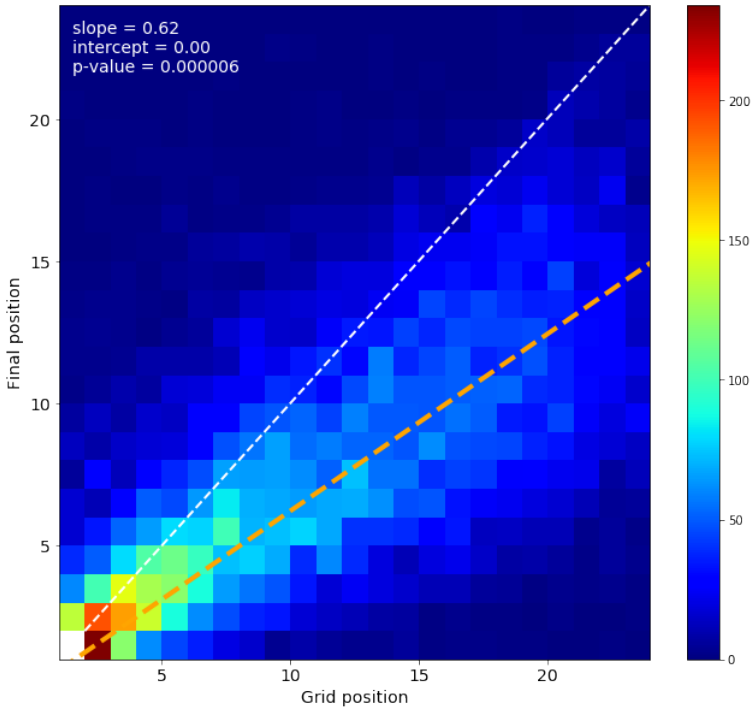

Figura 2: Posición final en relación a la posición inicial. El color muestra la cantidad de veces que un piloto ha ocupado cada posición inicial y final, según indica la barra de la derecha. El valor máximo implica que ese suceso ha ocurrido 400 o más veces. La línea discontinua blanca indica los puntos en los que un piloto empieza y termina en la misma posición. La línea discontinua amarilla nos indica la tendencia calculada por medio de un ajuste lineal. La caja de texto de la esquina superior izquierda muestra los resultados de este ajuste.

La figura 2 muestra este histograma. En la escala de color se han truncado los valores máximos a 400 para una mejor visibilidad del gráfico, ya que el rango es demasiado amplio. Sobreimpresa se muestra la línea de tendencia calculada a partir del ajuste lineal. Este ajuste arroja una pendiente de 0.62, con una ordenada en el origen forzada a pasar por cero, puesto que el ajuste mejora de esta forma.

Se observa que la tendencia es a ganar posiciones durante la carrera, lo cual sucede cerca del 63 % de las ocasiones.