Universidad
Católica del Norte

KOBE BRYANT SHOT SELECTION

Proyecto electivo de data science

Descripción breve

Kobe Bryant marked his retirement from the NBA by scoring 60 points in his final game as a Los Angeles Laker on Wednesday, April 12, 2016. Drafted into the NBA at the age of 17, Kobe earned the sport's highest accolades throughout his long career.

Juan Carlos Lamas Alfaro – Juan Pablo Martínez

Índice

Contenido

Ilustraciones

Contexto

Kobe Bryan se retiró de la NBA con un puntaje de 60 puntos en su último juego en “Los Angeles Laker” el miércoles 12 de abril del 2016. Ingresando a la NBA a la edad de 17 años, Kobe adquirió los más altos galardones de este deporte a lo largo de su carrera.

Usando 20 años de datos de los aciertos y fallas de Kobe, elegiremos el modelo que prediga mejor que tiros podría acertar.

KOBE BRYANT 1

Proceso de análisis

Primero que todo, se hace una mirada al “dataset”, para tener una idea general de cómo es el problema y como abordarlo.

```
df = pd.read_csv("data_proyecto.csv",index_col=0)
df.dropna(axis=0, how='any',inplace=True)
df.head()
```

action_type	combined_shot_type	game_event_id	game_id	lat	loc_x	loc_y	lon	minutes_remaining	period	playoffs	season	seconds_remaining
Jump Shot	Jump Shot	12	20000012	34.0443	-157	0	-118.4268	10	1	0	2000-01	22
Jump Shot	Jump Shot	35	20000012	33.9093	-101	135	-118.3708	7	1	0	2000-01	45
Jump Shot	Jump Shot	43	20000012	33.8693	138	175	-118.1318	6	1	0	2000-01	52
Driving Dunk Shot	Dunk	155	20000012	34.0443	0	0	-118.2698	6	2	0	2000-01	19
Jump Shot	Jump Shot	244	20000012	34.0553	-145	-11	-118.4148	9	3	0	2000-01	32

Analisis 1

```
df.tail()
```

action_type	combined_shot_type	game_event_id	game_id	lat	loc_x	loc_y	lon	minutes_remaining	period	playoffs	season	seconds_remaining
Driving Layup Shot	Layup	382	49900088	34.0443	0	0	-118.2698	7	4	1	1999-00	4
Jump Shot	Jump Shot	397	49900088	33.9963	1	48	-118.2688	6	4	1	1999-00	5
Running Jump Shot	Jump Shot	426	49900088	33.8783	-134	166	-118.4038	3	4	1	1999-00	28
Jump Shot	Jump Shot	448	49900088	33.7773	31	267	-118.2388	2	4	1	1999-00	10
Jump Shot	Jump Shot	471	49900088	33.9723	1	72	-118.2688	0	4	1	1999-00	39

Analisis 2

Posterior a ello, se hace una mirada más profunda a este.

```
df.describe()
```

	game_event_id	game_id	lat	loc_x	loc_y	lon	minutes_remaining	period	playoffs	seconds_remain
count	25697.000000	2.569700e+04	25697.000000	25697.000000	25697.000000	25697.000000	25697.000000	25697.000000	25697.000000	25697.000
mean	249.348679	2.474109e+07	33.953043	7.148422	91.257345	-118.262652	4.886796	2.520800	0.146243	28.311
std	149.778520	7.738108e+06	0.088152	110.073147	88.152106	0.110073	3.452475	1.151626	0.353356	17.523
min	2.000000	2.000001e+07	33.253300	-250.000000	-44.000000	-118.519800	0.000000	1.000000	0.000000	0.000
25%	111.000000	2.050006e+07	33.884300	-67.000000	4.000000	-118.336800	2.000000	1.000000	0.000000	13.000
50%	253.000000	2.090034e+07	33.970300	0.000000	74.000000	-118.269800	5.000000	3.000000	0.000000	28.000
75%	367.000000	2.960027e+07	34.040300	94.000000	160.000000	-118.175800	8.000000	3.000000	0.000000	43.000
max	653.000000	4.990009e+07	34.088300	248.000000	791.000000	-118.021800	11.000000	7.000000	1.000000	59.000

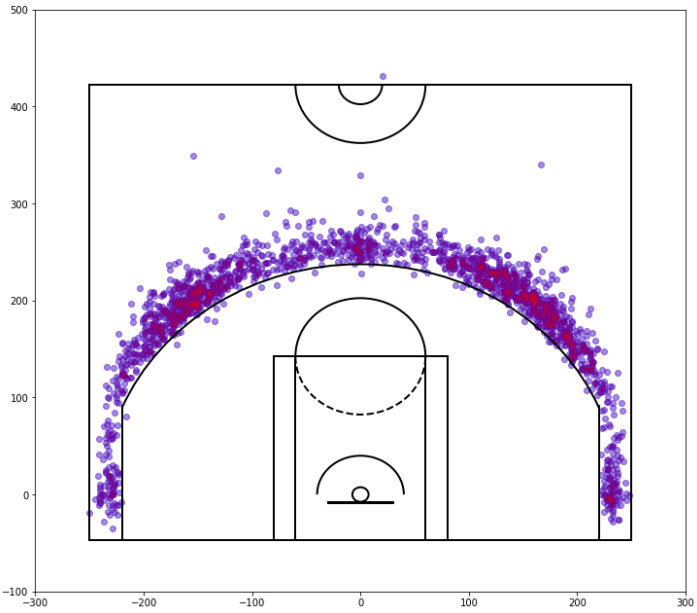
Analisis 3

Luego de un análisis de los datos y el contexto, se concluye que los tiros importantes, son los con objetivo de encestar, ya sea 2 o 3 puntos. Por lo que procedemos a filtrar aquellos datos que van enfocados a un tiro de 2 o 3 puntos y si este fue acertado o fallado.

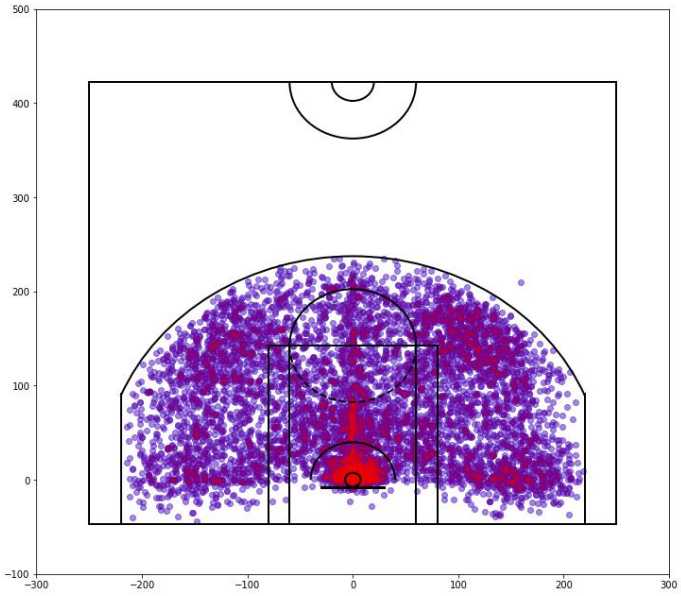
```
def points(shot):
    return shot


goal_df = df[df['shot_made_flag'].map(points) == 1.0]
twopts_df = goal_df[goal_df['shot_type'].map(points) == '2PT Field Goal']
threepts_df = goal_df[goal_df['shot_type'].map(points) == '3PT Field Goal']

fail_goal_df = df[df['shot_made_flag'].map(points) == 0]
fail_twopts_df = goal_df[goal_df['shot_type'].map(points) == '2PT Field Goal']
fail_threepts_df = goal_df[goal_df['shot_type'].map(points) == '3PT Field Goal']
```

Analisis 4

Cuyo resultado podemos apreciar en los siguientes renders sobre una chancha.

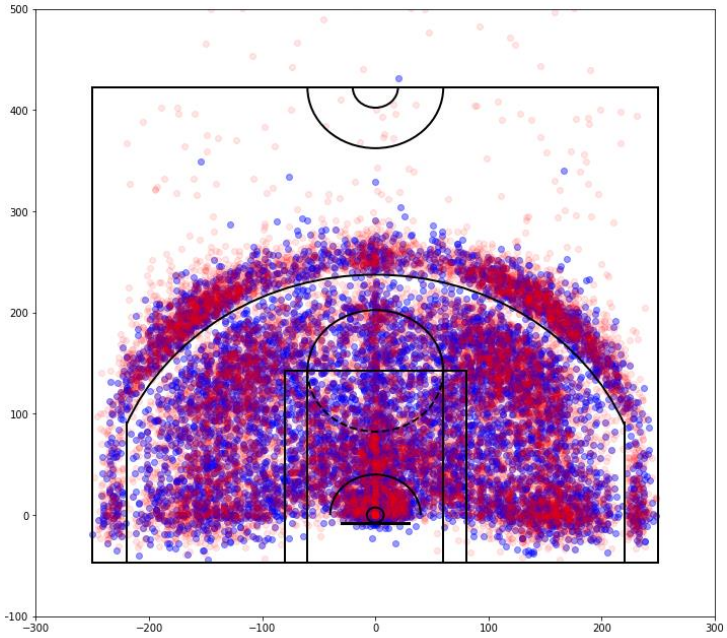


Analisis 5



Analisis 6

Y una vista general de estos tiros:

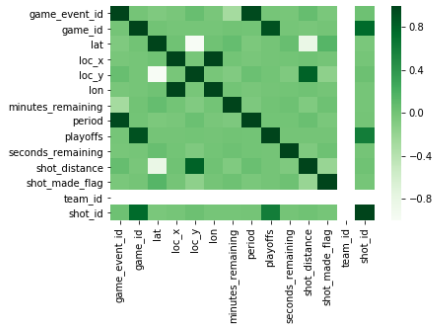


Analisis 7

Analizamos la relación entre cada una de las variables, obteniendo como resultado que variables inútiles para nuestro propósito

```
sns.heatmap(df.corr(), cmap="Greens")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2181e40aa58>
```

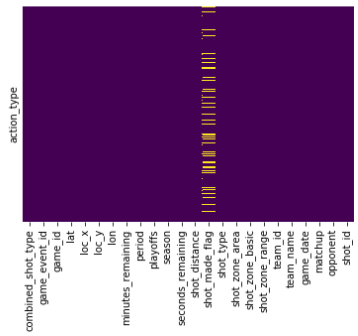


Analisis 8

Hacemos una limpieza de los datos, quitando columnas inservibles y datos nulos.

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2182217fa20>
```



Analisis 9

```
df = df.dropna()
df.drop('team_id',axis=1,inplace=True)
```

Analisis 10

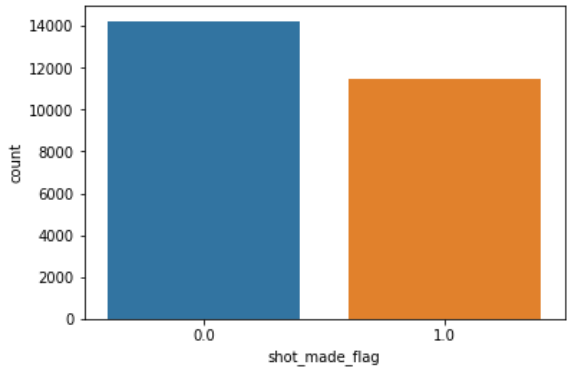
Vemos que columnas nos pueden ser útiles para crear nuestros modelos

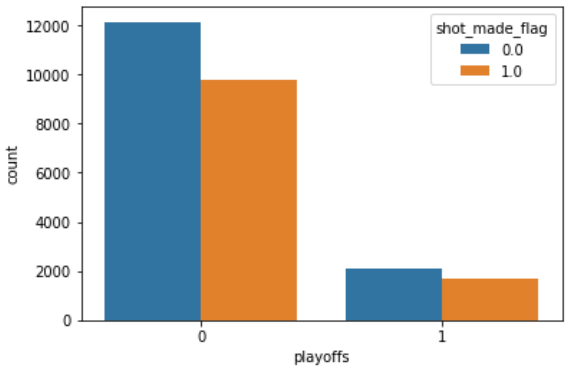
```
df.dtypes
```

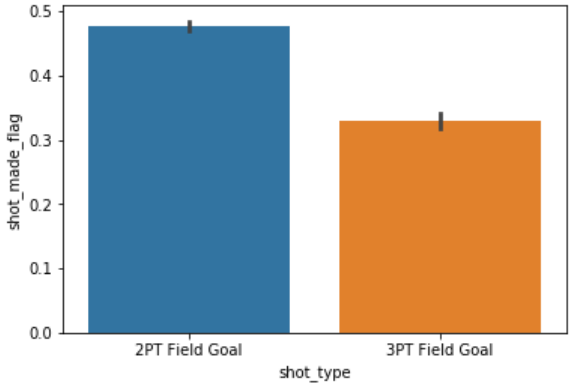
```
combined_shot_type     object
game_event_id           int64
game_id                 int64
lat                   float64
loc_x                   int64
loc_y                   int64
```

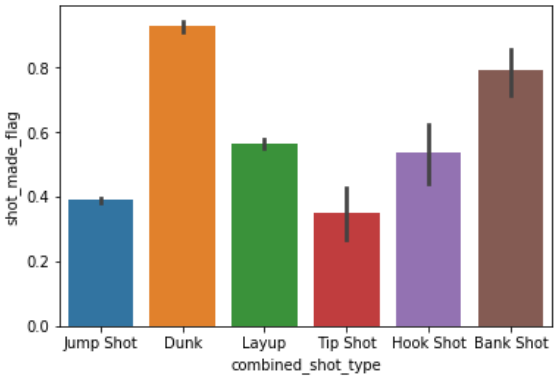
Analisis 11

Y procedemos a hacer comparaciones:









Desarrollo de los modelos

Modelo regresión logística

Con:

X=df[['period','minutes_remaining','playoffs','shot_distance','loc_x','loc_y','lat','lon','seconds_remaining']]

y = df['shot_made_flag']

Creamos y entrenamos el modelo, obteniendo como resultado:

	precision	recall	f1-score	support
0.0	0.61	0.75	0.67	4236
1.0	0.58	0.41	0.48	3474
micro avg	0.60	0.60	0.60	7710
macro avg	0.59	0.58	0.58	7710
weighted avg	0.60	0.60	0.59	7710

Regresión logistica 1

Modelo KNN

Normalizamos los datos:

	0	1	2	3	4	5	6	7	8
0	-1.320593	1.481054	-0.413876	0.164339	-1.491296	-1.035246	1.035246	-1.491296	-0.360186
1	-1.320593	0.612096	-0.413876	0.270852	-0.982533	0.496228	-0.496228	-0.982533	0.952371
2	-1.320593	0.322443	-0.413876	0.909929	1.188792	0.949998	-0.949998	1.188792	1.351845
3	-0.452239	0.322443	-0.413876	-1.433353	-0.064944	-1.035246	1.035246	-0.064944	-0.531389
4	0.416115	1.191401	-0.413876	0.057826	-1.382275	-1.160033	1.160033	-1.382275	0.210491

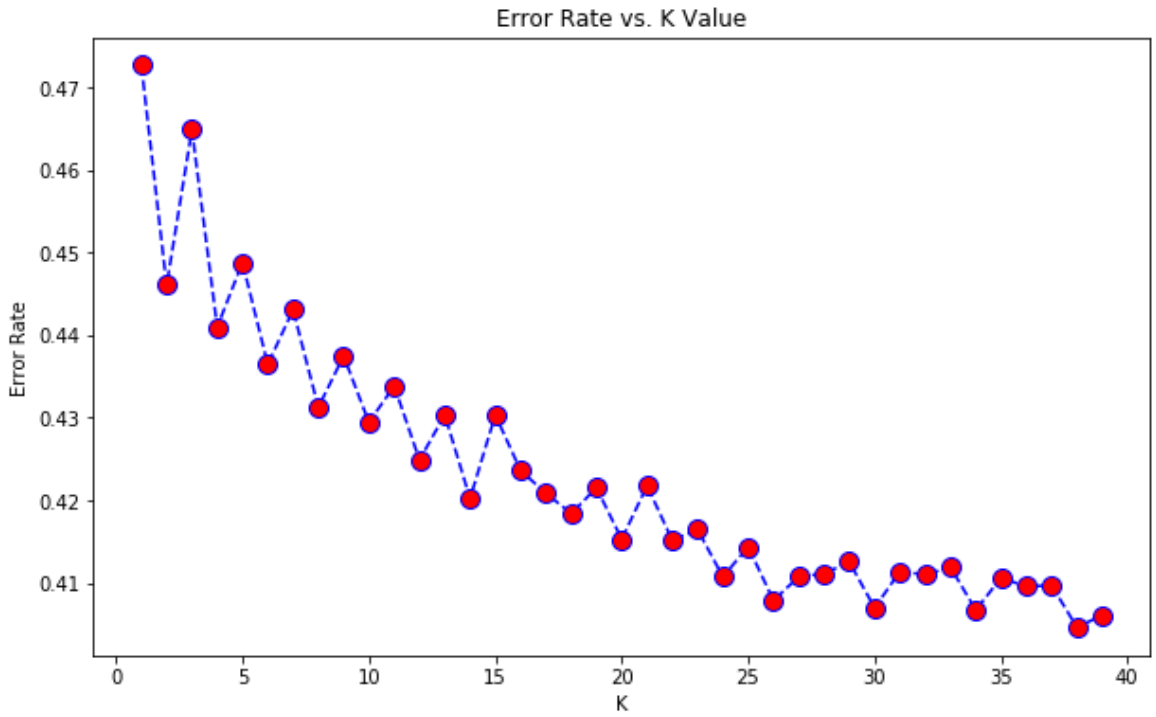
KNN 1

Con K igual a 1 obtenemos como resultado:

	precision	recall	f1-score	support
0.0	0.58	0.57	0.57	4296
1.0	0.47	0.47	0.47	3414
micro avg	0.53	0.53	0.53	7710
macro avg	0.52	0.52	0.52	7710
weighted avg	0.53	0.53	0.53	7710

KNN 2

Aplicando codo



KNN 3

Obtenemos como resultado:

	precision	recall	f1-score	support
0.0	0.61	0.76	0.68	4296
1.0	0.56	0.39	0.46	3414
micro avg	0.60	0.60	0.60	7710
macro avg	0.59	0.57	0.57	7710
weighted avg	0.59	0.60	0.58	7710

KNN 4

Random Forest

Resultado:

	precision	recall	f1-score	support
0.0	0.55	0.57	0.56	4264
1.0	0.44	0.42	0.43	3446
micro avg	0.50	0.50	0.50	7710
macro avg	0.49	0.49	0.49	7710
weighted avg	0.50	0.50	0.50	7710

Random forest 1

Conclusión

Luego de los análisis anteriormente realizados, podemos concluir que, a pesar de la poca relación entre las variables, usando un modelo de clasificación como “Regresión logística” se puede llegar a tener una predicción aceptable de si los tiros que hará serán acertados o no. Con una precisión de 61%.

Aplicación

```
df_p = df_prueba[pd.isnull(df_prueba['shot_made_flag'].map(points))]
```

```
X = df_p[['period','minutes_remaining','playoffs','shot_distance','loc_x','loc_y','lat','lon','seconds_remaining']]
y = df_p['shot_made_flag']

pred = logmodel.predict(X)

df_p['shot_made_flag'] = pred

df_p
```

Aplicación 1

	shot_distance	shot_made_flag	sh
	18	0.0	2
	2	1.0	2
	0	1.0	2
	0	1.0	2
	17	0.0	2
	20	0.0	2

Aplicación 2