Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Ingeniería en Ciencia de Datos y Matemáticas

Evidencia 4. Elemental, mi querido Watson

Análisis de Métodos de Razonamiento e Incertidumbre

Cantú Rodríguez Pamela A01285128

Díaz Seguy Laura Cecilia A01620523

Núñez López Daniel I. A01654137

Reyes Mancheno Paola Sofía A00831314

Torres Alcubilla María Fernanda A01285041

Supervisado por
Dr. Marco Otilio Peña Díaz

Monterrey, Nuevo León. Fecha, 17 de octubre de 2022

1. Problemática

La inteligencia artificial juega un gran papel en el desarrollo de agentes inteligentes que aprenden en ambientes de incertidumbre y que a través de su entrenamiento y razonamiento son capaces de tomar decisiones. Los conocimientos requeridos para llevar a cabo un buen despliegue de un agente inteligente es es saber sobre: creencias y deseos bajo incertidumbre, teoría utilitaria, redes de decisiones, técnicas de ML y decisiones complejas. Así mismo, es importante tener una base fuerte de lógica matemática.

Es por esto que saber diseñar agentes inteligentes puede llegar a ser útil para resolver tareas complejas que llevarían mucho tiempo o recursos para un humano. En este reporte se explorará el área de la inteligencia artificial (IA) enfocada en machine learning (ML) y los diversos algoritmos de clasificación que ayudan a clasificar datos con base en ciertas características con un fin específico.

Lo que se busca clasificar es una base de datos sobre los pasajeros del Titanic y sus características, y lo que se busca como resultado es saber si el pasajero sobreviviría al hundimiento del barco.

2. Propósito

El objetivo de este PBL fue reforzar los conocimientos del equipo sobre cómo resolver un problema de clasificación, esto al aplicar: técnicas conocidas de ML para hacer una predicción, el cálculo del coeficiente de Pearson para saber cuáles son las características más relevantes, limpieza de datos, cambios de variables cualitativas a categóricas, escalamiento de variables, etc.

3. Información

El dataset proporcionado consiste de dos partes; una parte para el entrenamiento del modelo, y otra con datos para probar el modelo entrenado. La parte de entrenamiento, además de mostrar las características de los pasajeros, nos enseña la clasificación correcta de cada uno.

La descripción de las variables es la siguiente:

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclasss	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	-
Age	Age in years	-
sibsp	# of siblings / spouses aboard the Titanic	-
parch	# of parents / children aboard the Titanic	-
ticket	Ticket number	-
fare	Passenger fare	-
cabin	Cabin number	-
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

Cuadro 1: Descripción de variables del dataset.

Notas:

1. pclass: funciona como proxy del estatus económico del pasajero.
2. age: es fraccionada si menor a 1. Si es estimación, está en la forma de xx.5
3. sibsp: se definen las relaciones de familia de la siguiente manera:
   a) Sibling = hermano, hermana, medio hermano, media hermana.
   b) Spouse = esposo, esposa (amantes y prometidos se ignoran).
4. parch: se definen las relaciones de familia de la siguiente manera (los niños que viajaban con una niñera fueron ignorados:
   a) Parent: madre, padre.
   b) Child: hija, hijo, hijastra, hijastro

El modelo a elegir es el de Random Tree Classifier. Este modelo se eligió debido a que toma en consideración varios arboles de decisión que clasifican a cada pasajero en si sobrevivió o no usando sus datos individuales. Al final, el modelo de Bosques Aleatorios clasifica al pasajero dependiendo de los resultados obtenidos de los arboles de decisión individuales.

Se utilizaron las librerías: numpy (para el álgebra lineal), pandas (para el procesamiento de datos), matplotlib (para las gráficas), sklearn (para el modelaje y escalamiento) y seaborn (para los mapas de calor).

4. Razonamiento

Se realizaron varias submissions con la finalidad de mejorar el modelo lo máximo posible, pero en este caso adjuntamos las 5 que nos devolvieron los resultados más favorecedores.

4.1. Submission 1

En la primera submission a exponer se usó como referencia el tutorial de Alexis Cook, en esta solamente se consideraron las características de 'Pclass', 'Sex', 'SibSp' y 'Parch' las cuales fueron, según nuestro criterio como equipo, las que tienen mayor importancia para comenzar el mejoramiento continuo del proyecto.

Posteriormente se creó y entrenó nuestro modelo con la función de RandomForestClassifier perteneciente a la librería sklearn, a este se le indicó tener 100 árboles de decisión, una profundidad máxima de 5 y para asegurar la reproducibilidad de los sets creados, se le asignó el valor 1 al *random_state*. Con esto, el modelo pudo obtener un resultado de 0.7751.

4.2. Submission 2

En esta propuesta se usa la prueba de correlación de Pearson con la finalidad de escoger las características más relevantes para el modelo, este prueba se muestra como mapa de color en la figura 1.En esta se destaca la relación negativa de la variable 'Pclass', la cual hace referencia a la clase perteneciente del boleto del pasajero, con la variable respuesta, 'Survived', se puede inferir que esto se debe a que los pasajeros con boletos de primera clase tenían mas probabilidades de sobrevivir que los pasajeros de tercera clase.

Por otro lado, la variable 'Fare', la cual se refiere al precio del boleto del pasajero, se correlaciona positivamente con la variable respuesta, 'Survived', lo que nos indica de manera similar que la relación pasada, que entre más elevado sea el precio del boleto que compró el pasajero, más probabilidades tuvo que sobrevivir

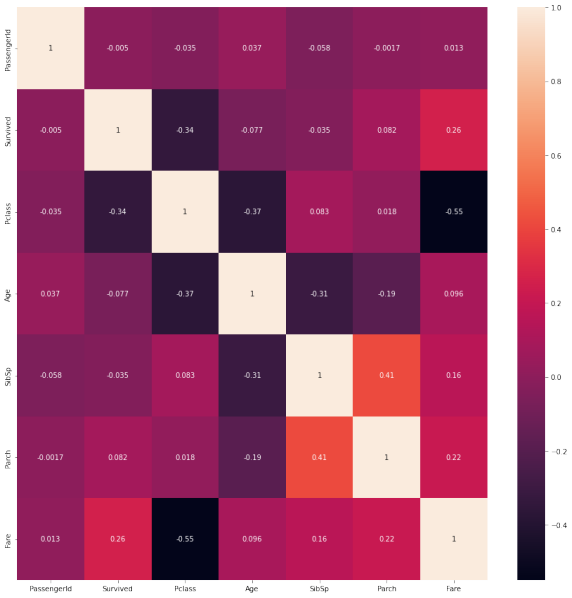

Figura 1: Mapa de calor.

Conociendo estas relaciones se decidió incluir la variable 'Fare' a nuestro modelo, es por esto que primero se manejó su único valor nulo. Se observó que este hacía referencia a un pasajero de 65 años de 3ra clase y para asignarle una tarifa similar a su perfil se calculó el promedio de las tarifas de los pasajeros mayores de 40 años y de 3ra clase.

Después de este llenado se creó y entrenó nuestro modelo de manera similar que en la Submission 1, con la diferencia de la agregación de la variable independiente 'Fare'. En esta propuesta se obtuvo un resultado ligeramente mejorado que al anterior, con un valor de 0.77751.

4.3. Submission 3

En esta propuesta se comenzó con el manejo de los valores nulos de las variables, para esto se calculó la cantidad de estos valores en los sets de entrenamiento y prueba. En ambos sets se encontró que las variables de 'Cabin' y 'Age' tenían una cantidad significante de datos nulos.

Para el manejo de valores nulos en 'Age' conociendo el contexto, donde los niños tenían una mayor probabilidad de salvarse, no se puede optar por el reemplazo con el promedio de edad de la base de datos. Es por esto que, basándonos en el código propuesto por el kaggler Corazon17, a partir de la variable 'Name' que hace referencia al 'título' del pasajero se realizó una agrupación y clasificación para obtener la edad promedio por cada título y poder reemplazarla en los valores faltantes.

En cuanto la variable 'Cabin', esta contaba con una gran cantidad de valores nulos los cuales no podían ser imputados por otra variable como en el caso de 'Age', es por esto que se decidió no tomarla en cuenta para nuestro modelo.

Posterior al manejo de datos nulos se realizó otra prueba de correlación con el objetivo de observar si alguna correlación mostraba algún cambio, el mapa de calor mostrado en la figura 2 nos indica un cambio de las relaciones de la variable 'Age', es por esto que esta fue agregada al modelo.

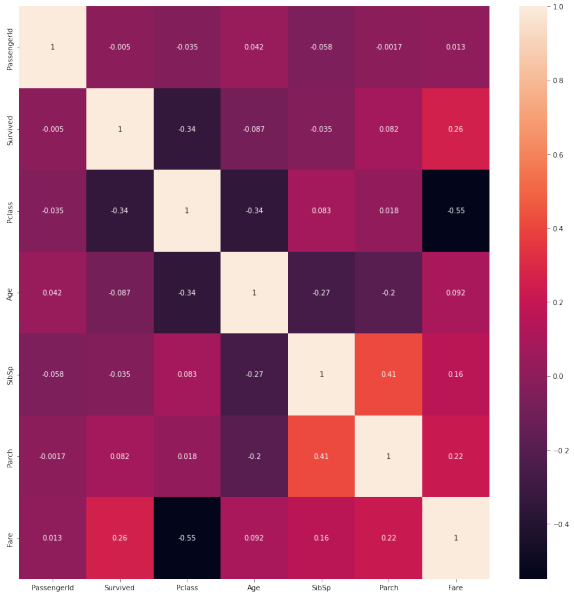

Figura 2: Mapa de calor con manejo de valores nulos.

Después de crear y entrenar nuevamente el modelo con los mismos argumentos que en las propuestas pasadas, con esta adición en nuestras variables independientes se obtuvo un resultado de 0.77990.

4.4. Submission 4

Como no se observa una gran mejora en nuestros resultados, en esta propuesta se decide cambiar el enfoque pasado de agregar variables a la transformación de variables categóricas y escalamiento.

Primero se eliminaron las variables de 'Ticket' y 'PassengerId' ya que no aportaban información y después las variables de 'Sex', 'Embarked' y 'Title' se convirtieron a *'variables dummies'* para que pudieran ser utilizadas en el modelo.

Para el escalamiento se aplicó la estandarización en ambos sets de datos, con estos datos se creó

y entrenó el modelo obteniendo un resultado de 0.7790, lo cual no representa una mejora.

4.5. Submission 5

Debido a que la submission 4 no obtuvo resultados tan eficientes como la submission 3, en este intento de mejora se toma como referencia la submission 3. A partir de este modelo se realizó una modificación de parámetros y features del modelo base, en donde se incrementó un 50 % el número de árboles de decisiones del modelo y se eliminó la variable independiente 'Sex'.

A partir de los cambios mencionados anteriormente se logró llegar a un score de 0.77511, que en este caso fue a lo mejor que pudimos llegar modificando el modelo de la sumbission de referencia.

5. Conclusiones

En las primeras submissions pudimos observar un mínimo aumento entre los resultados y al modificar parámetros o variables a considerar en nuestro mejor modelo no pudimos mejorar este score. Es por esto que consideramos que el manejo de valores nulos y la observación de las correlaciones entre las variables es lo que más ayuda para el mejoramiento del score, como se pudo observar en la submission 3, el cuál consideramos nuestro mejor modelo con un score de 0.77990.

Un punto de mejora sería realizar diversas pruebas con diferentes algoritmos de clasificación para comprar los scores y observar si se puede presentar un mejor modelo para las predicciones de la supervivencia del Titanic.

6. Apéndice: Código en Python

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv
```

Load the data

```
train_data = pd.read_csv("/kaggle/input/titanic/train.csv")
train_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
test_data = pd.read_csv("/kaggle/input/titanic/test.csv")
test_data.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female							

```
    4   22.0      1      1   3101298   12.2875   NaN        S
```

Modelo elegido

Se utilizará el modelo de aprendizaje automatico de Bosques Aleatorios. Este modelo se eligió debido a que toma en consideracion varios arboles de decisión que clasifican a cada pasajero en si sobrevivió o no usando sus datos individuales. Al final, el modelo de Bosques Aleatorios clasifica al pasajero dependiendo de los resultados obtenidos de los arboles de decisión individuales.

Submission 1

Usando como referencia el tutorial realizado por Alexis Cook, la primera propuesta a entregar toma en cuenta las caracteristicas: "Pclass", "Sex", "SibSp", "Parch". Estas caracteristicas son seleccionadas basadas solamente en intuición y sin usar un modelo de selección de caracteristicas previo.

El resultado de esta propuesta es 0.7751. El resultado irá subiendo conforme se apliquen tecnicas de limpieza de datos y cross-validation.

```
from sklearn.ensemble import RandomForestClassifier

y = train_data["Survived"]

features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_data[features])

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")
```

```
Your submission was successfully saved!
```

Submission 2

La segunda propuesta a entregar usará una prueba de correlación de Pearson con la finalidad de elegir las caracteristicas mas relevantes para el modelo.

En el mapa de calor proporcionado, se puede observar que la variable "Pclass", la cual se refiere a la clase a la que pertenecía el boleto del pasajero, tiene una correlación negativa con la variable respuesta. Se puede inferir que esto se debe a que los pasajeros con boletos de primera clase tenian mas probabilidades de sobrevivir a los pasajeros de tercera clase.

Por otro lado, la variable "Fare", la cual se refiere al precio del boleto del pasajero, se correlaciona positivamente con la variable respuesta. De manera similar a la variable "Pclass", esto se refiere a que entre mas caro sea el boleto que compró el pasajero, mas probabilidades tuvo de sobrevivir.

La segunda propuesta agrega "Fare" al modelo anterior. Da un total de 0.77751

```
import seaborn as sns
fig, ax = plt.subplots(figsize=(15,15))
sns.heatmap(train_data.corr(method= 'pearson'), annot=True, ax=ax)
```

```
<AxesSubplot:>
```

```
[]
```

```
# Manejar valores nulos para la columna "Fare" en test_data
# Poner un precio similar al de alguien en su misma clase y de su misma edad
test_data[test_data["Fare"].isnull()]
```

```
     PassengerId  Pclass                Name   Sex   Age  SibSp  Parch Ticket  \
152         1044       3  Storey, Mr. Thomas  male  60.5      0      0   3701

     Fare Cabin Embarked
152   NaN   NaN        S
```

```
# Le pondremos el promedio de fare de personas arriba de
40 a os y que estan en la misma clase 3
third_class_over_40_df = test_data[(test_data["Pclass"] == 3)
& (test_data["Age"] > 40)]
mean_fare = third_class_over_40_df["Fare"].mean()
mean_fare
```

```
10.525585714285715
```

```
test_data["Fare"] = test_data["Fare"].fillna(mean_fare)
```

```
from sklearn.ensemble import RandomForestClassifier

y = train_data["Survived"]

features = ["Pclass", "Sex", "SibSp","Parch","Fare"]
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_data[features])

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")
```

```
Your submission was successfully saved!
```

Submission 3

En esta tercera entrega se lidiará con los valores nulos en ambos datasets, posteriormente realizar otra prueba de correlación para ver si algo cambia en las variables que muestran muchos valores nulos.

El resultado es 0.77990.

```
train_data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
test_data.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          327
Embarked         0
dtype: int64
```

En ambos datasets, las variables "Age" y "Cabin" muestran valores nulos.

Age

La variable de la edad de cada pasajero no puede ser imputada solamente con el promedio de la edad de los pasajeros, la cual es 29. En el caso del accidente del Titanic, se sabe que los ni os tenian una mayor probabilidad de salvarse por lo tanto debemos poner una edad adecuada para cada pasajero.

Para hacer una imputación adecuada, nos basaremos en el codigo propuesto por el kaggler Corazon17. Este usa la variable "Name", la cual contiene el 'titulo' de cada pasajero (Mister, Mrs, Master, etc.). A traves de clasificar a los pasajeros segun su titulo, podemos imputar valores promedio por cada titulo.

```
# Extraemos titulo de cada pasajero
train_data["Title"] = train_data["Name"].str.extract('([A-Za-z]+)\.')
test_data["Title"] = test_data["Name"].str.extract('([A-Za-z]+)\.')
train_data["Title"].value_counts()
```

```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
```

```
Rev            6
Mlle           2
Major          2
Col            2
Countess       1
Capt           1
Ms             1
Sir            1
Lady           1
Mme            1
Don            1
Jonkheer       1
Name: Title, dtype: int64
```

```
# Agrupamos
def convert_title(title):
    if title in ["Ms", "Mile", "Miss"]:
        return "Miss"
    elif title in ["Mme", "Mrs"]:
        return "Mrs"
    elif title == "Mr":
        return "Mr"
    elif title == "Master":
        return "Master"
    else:
        return "Other"

train_data["Title"] = train_data["Title"].map(convert_title)
test_data["Title"] = test_data["Title"].map(convert_title)

train_data["Title"].value_counts()
```

```
Mr         517
Miss       183
Mrs        126
Master      40
Other       25
Name: Title, dtype: int64
```

```
# Obtenemos el promedio de la edad de cada titulo
train_data.groupby('Title')['Age'].mean()
```

```
Title
Master     4.574167
Miss      21.816327
Mr        32.368090
Mrs       35.788991
Other     43.750000
Name: Age, dtype: float64
```

```
# Imputamos estos valores para cada titulo
data = [train_data, test_data]
for df in data:
    df.loc[(df["Age"].isnull()) & (df["Title"]=='Master'), 'Age'] = 5
    df.loc[(df["Age"].isnull()) & (df["Title"]=='Miss'), 'Age'] = 22
    df.loc[(df["Age"].isnull()) & (df["Title"]=='Mr'), 'Age'] = 32
    df.loc[(df["Age"].isnull()) & (df["Title"]=='Mrs'), 'Age'] = 36
    df.loc[(df["Age"].isnull()) & (df["Title"]=='Other'), 'Age'] = 44
```

```
    # Eliminamos "Name" de ambos df
    train_data.drop("Name", axis=1, inplace=True)
    test_data.drop("Name", axis=1, inplace=True)
```

Cabin

Debido a que esta variable muestra demasiados valores nulos y que no pueden ser imputados por alguna otra variable en el df, se decide no tomarla en cuenta en el modelo.

```
    train_data.drop("Cabin", axis=1, inplace=True)
    test_data.drop("Cabin", axis=1, inplace=True)
```

Prueba de correlación sin valores nulos

Agregamos "Age" como caracteristica a considerar en el modelo.

```
    import seaborn as sns
    fig, ax = plt.subplots(figsize=(15,15))
    sns.heatmap(train_data.corr(method= 'pearson'), annot=True, ax=ax)
```

```
    <AxesSubplot:>
```

```
[]
```

```
    from sklearn.ensemble import RandomForestClassifier

    y = train_data["Survived"]

    features = ["Pclass","Age","Sex","SibSp","Parch","Fare"]
    X = pd.get_dummies(train_data[features])
    X_test = pd.get_dummies(test_data[features])

    model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
    model.fit(X, y)
    predictions = model.predict(X_test)

    output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})
    output.to_csv('submission.csv', index=False)
    print("Your submission was successfully saved!")
```

```
    Your submission was successfully saved!
```

Submission 4

Debido a que el score no está subiendo drasticamente al agregar caracteristicas al modelo, cambiaremos las variables categoricas para que puedan ser usadas en el modelo. Posteriormente, haremos un escalamiento de variables.

El resultado fue 0.77990

```
    # Las variables categoricas son "Sex", "Embarked", "Title" y "Ticket"
    train_data.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 891 entries, 0 to 890
```

```
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Sex          891 non-null    object
 4   Age          891 non-null    float64
 5   SibSp        891 non-null    int64
 6   Parch        891 non-null    int64
 7   Ticket       891 non-null    object
 8   Fare         891 non-null    float64
 9   Embarked     889 non-null    object
 10  Title        891 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB

# La variable "Ticket" no da informacion valiosa al modelo y por ende es eliminada.
train_data.drop("Ticket", axis=1, inplace=True)
test_data.drop("Ticket", axis=1, inplace=True)

# La variable "PassengerId" igualmente es eliminada
train_data.drop("PassengerId", axis=1, inplace=True)
test_data.drop("PassengerId", axis=1, inplace=True)

# Las variables son codificadas para que puedan ser usadas en el modelo
train_data = pd.get_dummies(train_data, prefix=["Sex", "Embarked", "Title"])
test_data = pd.get_dummies(test_data, prefix=["Sex", "Embarked", "Title"])

# Dividimos el dataset
y = train_data["Survived"]
X = train_data.drop("Survived", axis=1)
X_test = test_data.copy()

# Realizamos un escalamiento de variables
X_train_scaled = StandardScaler().fit_transform(X)
X_test_scaled = StandardScaler().fit_transform(X_test)

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X_train_scaled, y)
predictions = model.predict(X_test_scaled)

output = pd.DataFrame({'Survived': predictions}, index=test_data.index+1)
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")

Your submission was successfully saved!
```

Submission 5

Debido a que el modelo anterior dió un score mas bajo, utilizaremos el modelo del Submission 3 y modificaremos los hiperparametros del modelo, al igual que modificar unos cuantos features utilizados.

Score: 0.77511

```
from sklearn.ensemble import RandomForestClassifier

y = train_data["Survived"]

features = ["Pclass","Age","SibSp","Parch","Fare"]
#X = pd.get_dummies(train_data[features])
#X_test = pd.get_dummies(test_data[features])

model = RandomForestClassifier(n_estimators=150, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

output = pd.DataFrame({'Survived': predictions}, index=test_data.index+1)
output.to_csv('submission.csv', index=False)
print("Your submission was successfully saved!")

Your submission was successfully saved!
```