Procedemos a realizar este ejercicio:

Ejercicio práctico en grupos de 4 personas. En los casos que el grupo sea de 3 personas, el Developer_1, hará también de Developer_4 con las ramas específicas de Developer_4

En este ejercicio vamos a crear un fichero app.py donde vamos a realizar una implantación en producción con una visualización de los datos y un modelado.
Todo ello con los datos del clásico para un estudiante de Data Science como es el **Titanic**
(NO OS PREOCUPEIS, NO TENEIS QUE SABER LO QUE HACEIS, YA LO APRENDEREIS)

En este ejercicio, realizaremos una serie de tareas que nos ayudarán a afianzar lo explicado en el taller.

Para ello, formaremos grupos de 4 personas que representarán cada uno de los miembros del equipo: Developer_1, Developer_2, Developer_3, Developer_4. Cada uno de estos miembros será elegido por consenso del grupo y tendrá un papel determinado.

Deberéis seguir los pasos indicados uno por uno sin saltaros nada para que el resultado sea un proyecto funcional.

No se pedirá desarrollo de código, se os proporcionará una parte a cada uno para realizar las tareas.

En algunos momentos podréis tener algún conflicto con parte de los códigos que habéis escrito. Tendréis que evaluar el conflicto y quedaros con la parte de código que consideréis correcta (dado que vamos a utilizar librerías que aún no habéis utilizado, no dudéis en preguntar a los profesores, aunque los conflictos hemos intentado que sean obvios para que no tengáis problemas).

Cuando terminéis de escribir el código, seguid las indicaciones del final.

Developer_1:

1. Crea un repositorio vacío en **Github**
2. Da permisos de push a Developer_2, Developer_3 y Developer_4 en **Github**
3. Crea desde **git bash (o Visual Studio Code)** la develop-branch que será la rama que reciba los cambios que se produzcan en las epic branches de los tres Developeres
4. **NO OLVIDES HACER git push origin develop-branch**
5. Comprueba que en **Github** están ambas ramas [main, develop-branch]
6. **AHORA EL RESTO DE DEVELOPERS PUEDEN EMPEZAR A TRABAJAR, NO ANTES**
7. Crea la epic-branch-1
   En ella te encargarás de desarrollar [app.py]
   Para ello **partiendo de epic-branch-1** deberás crear una rama ticket por cada uno de los trozos de código que añadas a [app.py]

   - Crea la rama branch-ticket-1 para añadir el **Trozo de código 1**. Cuando lo tengas, fusiona con la epic-branch-1
   - Crea la rama branch-ticket-2 para añadir **Trozo de código 2**. Cuando lo tengas, fusiona con la epic-branch-1.
   - Crea la rama branch-ticket-3 a partir de epic-branch-1. Después, crea un sub-branch dentro de branch-ticket-3, la branch-ticket-3-1. Inserta aquí el trozo de código 3. Fusiona primero branch-ticket-3 con branch-ticket-3-1.
   - Vuelve a epic-branch-1. Inserta este código, añade y commit. Fusiona con branch-ticket-3
   - Asegúrate de que has fusionado todas tus branches ticket con la epic-branch-1 y que tu archivo app.py es idéntico al del repo original. Si no es así, haz una rama ticket para cada cosa que añadas.
   - Es el momento en el que debes mandar tu trabajo al repositorio. usa git push origin epic-branch-1 para esto.

código developer_1:

```
# Developer 1 - Parte 1

'''
Importamos librerías
'''
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Developer 1 - Parte 2

'''
Importamos librerías
'''
import pandas
import numpy
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
st.set_option('deprecation.showPyplotGlobalUse', False)

# Developer 1 - Parte 3

'''
Cargamos los datos
'''
train_df = pd.read_csv('data/train.csv')
test_df = pd.read_csv('data/test.csv')
```

Developer_2

1. Clona el repositorio Creado por Developer_1 en local.
2. Al clonarte el repositorio, solo habrás traído la rama main, sin embargo, necesitarás la rama develop-branch para continuar trabajando.
3. Antes de continuar, asegúrate de que Developer_1 ha creado la rama develop y la ha pusheado a Github, Entonces sigue los siguientes pasos https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git
4. Crea la epic-branch-2 desde develop-branch.
    1. Para esto deberás crear, partiendo de epic-branch-2 una rama ticket por cada uno de los trozos de código que añadas a app.py
    2. Crea la rama branch-ticket-1 para añadir **Trozo de código 1**. Cuando lo tengas, fusiona con la epic-branch-2.
    3. Crea la rama branch-ticket-2 para añadir **Trozo de código 2.** Cuando lo tengas, fusiona con la epic-branch-2
    4. Crea la rama branch-ticket-3 a partir de epic-branch-2. Después, crea dos sub-branches dentro de branch-ticket-3, la branch-ticket-3-1 y la branch-ticket-3-2. Inserta en cada una de ellas el trozo de código 3 y el 4, respectivamente. Fusiona primero branch-ticket-3 con branch-ticket-3-1. Luego branch-ticket-3 con branch-ticket-3-2.
    5. Vuelve a epic-branch-2. Inserta este código, añade y commit. Fusiona con branch-ticket-3
5. Resuelve los conflictos que te haya podido surgir y quédate con el trozo de código de la rama branch-ticket-3.
6. Asegúrate de que has fusionado todas tus branches con epic-branch-2 y que tu parte del archivo app.py es idéntico al del repo original. Si no es así, haz una rama ticket para cada cosa que añadas.
7. **ES EL MOMENTO:** manda tu trabajo al repositorio utilizando git push origin epic-branch-2 para esto.

código developer_2:

```
# Developer 2 - Parte 1

'''
Hacemos Featuring Engineering
'''
train_df['Sex'] = train_df['Sex'].map({'male': 0, 'female': 1})
test_df['Sex'] = test_df['Sex'].map({'male': 0, 'female': 1})
train_df['Sex'].fillna(train_df['Age'].mean(), inplace=True)
test_df['Sex'].fillna(test_df['Age'].mean(), inplace=True)

# Developer 2 - Parte 2

'''
Hacemos Featuring Engineering
'''
train_df['Age'].fillna(train_df['Age'].mean(), inplace=True)
test_df['Age'].fillna(test_df['Age'].mean(), inplace=True)
train_df['Embarked'].fillna(train_df['Embarked'].mode()[0], inplace=True)
test_df['Embarked'].fillna(test_df['Embarked'].mode()[0], inplace=True)

# Developer 2 - Parte 3

'''
Hacemos Featuring Engineering
'''
train_df['CabinBool'] = (train_df['Cabin'].notnull().astype('int'))
test_df['CabinBool'] = (test_df['Cabin'].notnull().astype('int'))
train_df = pd.concat([train_df, pd.get_dummies(train_df['Embarked'],
prefix='Embarked')], axis=0)
test_df = pd.concat([test_df, pd.get_dummies(test_df['Embarked'], prefix='Embarked')],
axis=0)


# Developer 2 - Parte 4

'''
Hacemos Featuring Engineering
'''
train_df = pd.concat([train_df, pd.get_dummies(train_df['Embarked'],
prefix='Embarked')], axis=1)
test_df = pd.concat([test_df, pd.get_dummies(test_df['Embarked'], prefix='Embarked')],
axis=1)
test_df['Fare'].fillna(test_df['Fare'].dropna().mean(), inplace=True)
```

Developer_3

1. Clona el repositorio creado por Developer_1 en local.
2. Al clonarte el repositorio, solo habrás traído la rama main, sin embargo, necesitarás la rama develop-branch para continuar trabajando.
3. Antes de continuar, asegúrate de que Developer_1 ha creado la rama develop y la ha pusheado a Github. Entonces sigue los siguientes pasos: https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git
4. Crea la epic-branch-3 desde develop-branch en esta te encargarás de desarrollar tu parte de app.py.
    1. Crea rama branch-ticket-1 para añadir **Trozo de código 1.** Cuando lo tengas, fusiona con la epic-branch-3
    2. Crea rama branch-ticket-2 para añadir **Trozo de código 2**. Cuando lo tengas, fusiona con la epic-branch-3
    3. Crea rama branch-ticket-3 a partir de epic-branch-3. Después, crea dos sub-branches dentro de branch-ticket-3, la branch-ticket-3-1 y la branch-ticket-3-2. Inserta en cada una de ellas el trozo de código 3 y el 4, respectivamente. Fusiona primero branch-ticket-3 con branch-ticket-3-1. Luego branch-ticket-3 con branch-ticket-3-2.
    4. Vuelve a epic-branch-3
    5. Inserta este código, añade y commit. Fusiona con branch-ticket-3
5. Resuelve los conflictos que puedan surgir y quédate con el trozo de código de la rama branch-ticket-3.
6. Asegúrate de que has fusionado todas tus branches ticket con la epic-branch-3y que tu parte del archivo app.py es **idéntico al del repo original.**
    1. Si no es así, haz una rama ticket para cada cosa que añadas.
7. Es el momento en el que debes mandar tu trabajo al repositorio, usa git push origin epic-branch-3para esto.

código developer_3:

```
# Developer 3 - Parte 1

'''
Generamos Variables dependientes e independientes
'''
variables=['Pclass',  'Age', 'Sex', 'SibSp', 'Parch', 'Fare', 'CabinBool', 'Embarked_C',
'Embarked_S', 'Embarked_Q']
X = train_df[variables]
y = train_df['Survived']

# Developer 3 - Parte 2

'''
Creamos el modelo
'''
X = train_df[['Pclass',  'Age', 'Sex', 'SibSp', 'Parch', 'Fare', 'CabinBool', 'Embarked_C',
'Embarked_S', 'Embarked_Q']]
y = train_df[['Survived']]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=0)
model = RandomForestClassifier(n_estimators=100, random_state=0)
model.fit(X_train, y_train)

# Developer 3 - Parte 3

'''Hacemos predicciones'''
model.fit(X_train, y_val)
y_pred = model.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
importances =
pd.DataFrame({'feature':X_train.columns,'importance':np.round(model.feature_importan
ces_,3)})
importances =
importances.sort_values('importance',ascending=False).set_index('feature')

# Developer 3 - Parte 4

'''
Visualizamos la variable Age
'''
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80]
train_df['AgeGroup'] = pd.cut(train_df['Age'], bins)
survived_age = train_df[train_df['Survived']==1]['AgeGroup'].value_counts(sort=False)
dead_age = train_df[train_df['Survived']==0]['AgeGroup'].value_counts(sort=False)
age_df = pd.DataFrame([survived_age,dead_age],index=['Survived','Dead'])
age_df.plot(kind='bar', stacked=True)
plt.xlabel('Age Group')
plt.ylabel('Number of passengers')
plt.title('Distribution of passengers by age and survival')
st.pyplot()
```

Developer_4

1. Clona el repositorio creado por Developer_1 en local.
2. Al clonarte el repositorio, solo habrás traído la rama main, sin embargo, necesitarás la rama develop-branch para continuar trabajando.
3. Antes de continuar, asegúrate de que Developer_1 ha creado la rama develop y la ha pusheado a Github. Entonces sigue los siguientes pasos: https://stackoverflow.com/questions/67699/how-to-clone-all-remote-branches-in-git
4. Crea la epic-branch-4 desde develop-branch en esta te encargarás de desarrollar tu parte de app.py.
    1. Crea rama branch-ticket-1 para añadir **Trozo de código 1.** Cuando lo tengas, fusiona con la epic-branch-4
    2. Crea rama branch-ticket-2 para añadir **Trozo de código 2**. Cuando lo tengas, fusiona con la epic-branch-4
    3. Crea rama branch-ticket-3 a partir de epic-branch-4. Después, crea dos sub-branches dentro de branch-ticket-3, la branch-ticket-3-1 y la branch-ticket-3-2. Inserta en cada una de ellas el trozo de código 3 y el 4, respectivamente. Fusiona primero branch-ticket-3 con branch-ticket-3-1. Luego branch-ticket-3 con branch-ticket-3-2.
    4. Vuelve a epic-branch-4
    5. Inserta este código, añade y commit. Fusiona con branch-ticket-3
5. Resuelve los conflictos que puedan surgir y quédate con el trozo de código de la rama branch-ticket-3.
6. Asegúrate de que has fusionado todas tus branches ticket con la epic-branch-4 y que tu parte del archivo app.py es **idéntico al del repo original.**
    1. Si no es así, haz una rama ticket para cada cosa que añadas.

Es el momento en el que debes mandar tu trabajo al repositorio, usa git push origin epic-branch-4 para esto.

código developer_4:

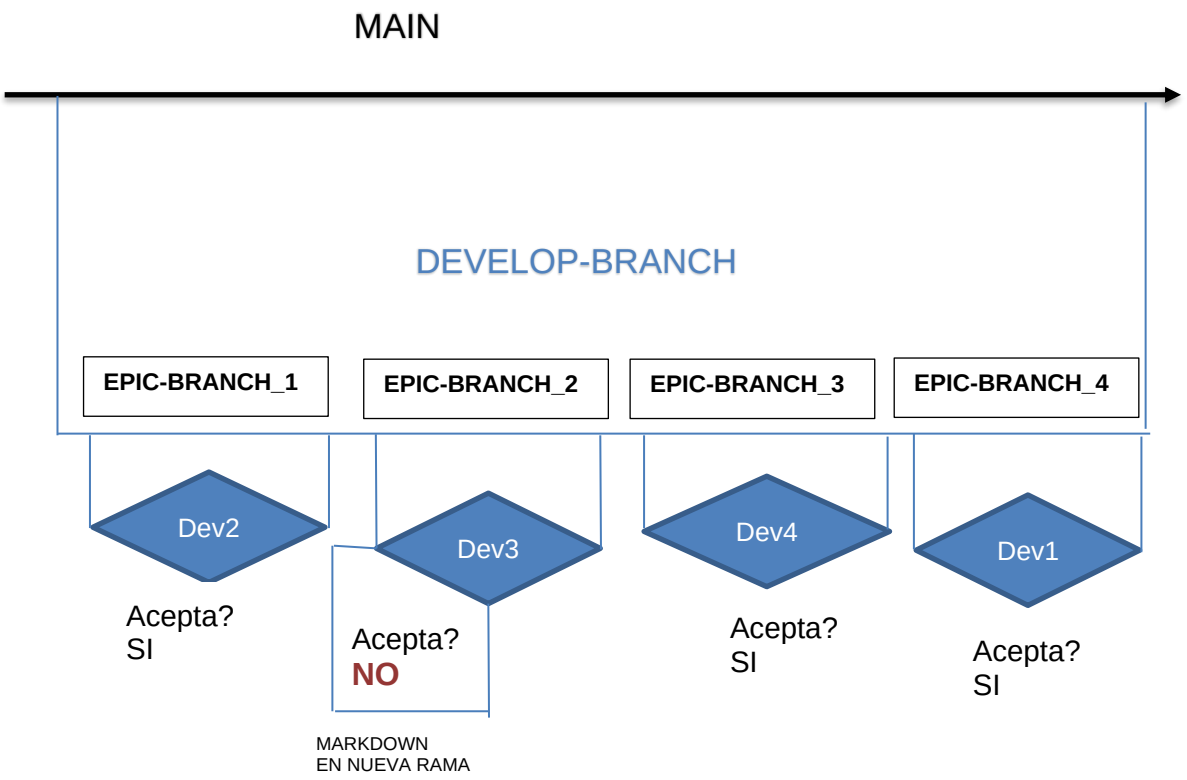
```
# Developer 4 - Parte 1

'''
Visualizamos la variable Clase
'''
survived_class = train_df[train_df['Survived']==1]['Pclass'].value_counts(sort=False)
dead_class = train_df[train_df['Survived']==0]['Pclass'].value_counts(sort=False)
class_df = pd.DataFrame([survived_class,dead_class],index=['Survived','Dead'])
class_df.plot(kind='bar', stacked=True)
plt.xlabel('Class')
plt.ylabel('Number of passengers')
plt.title('Survival rate by class')
st.pyplot()

# Developer 4 - Parte 2

'''
Visualizamos la variable Sexo
'''

survived_sex = train_df[train_df['Survived']==1]['Sex'].value_counts(sort=False)
dead_sex = train_df[train_df['Survived']==0]['Sex'].value_counts(sort=False)
sex_df = pd.DataFrame([survived_sex,dead_sex],index=['Survived','Dead'])
sex_df.plot(kind='bar', stacked=True)
plt.xlabel('Sex')
plt.ylabel('Number of passengers')
plt.title('Survival rate by sex')
st.pyplot()

# Developer 4 - Parte 3

'''
Visualizamos la variable Embarked
'''

survived_embark =
train_df[train_df['Survived']==1]['Embarked'].value_counts(sort=False)
dead_embark = train_df[train_df['Survived']==0]['Embarked'].value_counts(sort=False)
embark_df =
pd.DataFrame([survived_embark,dead_embark],index=['Survived','Dead'])
embark_df.plot(kind='bar', stacked=True)
plt.xlabel('Embarked')
plt.ylabel('Number of passengers')
plt.title('Survival rate by port of embarkation')
st.pyplot()
```

 # Developer 4 - Parte 4

'''
Visualizamos la importancia de las variables
'''
survived_embark =
train_df[train_df['Survived']==1]['Embarked'].value_counts(sort=True)
dead_embark = train_df[train_df['Survived']==0]['Embarked'].value_counts(sort=False)
st.write('Visualización del modelo y datos')
st.write('Importancia de características en el modelo')
st.bar_chart(importances)

LLEGADO A ESTE PUNTO, ES IMPORTANTE QUE TODOS LOS MIEMBROS DEL EQUIPO ESTÉIS MÁS O MENOS EN EL MISMO PUNTO, SI NO ES ASÍ, ES TAMBIÉN TU RESPONSABILIDAD QUE ESO SEA ASÍ. ÉCHALES UNA MANO!!!

Vamos a realizar las funciones de esta imagen



¿Qué modificación hay que hacer en Epic-Branch_2?

Agrega una nueva rama, donde crearás un Markdown con los nombres, apellidos y e-mails de los integrantes del desarrollo.

¡¡Ya habéis terminado!! Ahora puedes hacer git pull desde develop-branch (desde tu ordenador) y ya tendrás el proyecto acabado.