UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

título del TFG
Documentación Técnica

Presentado por nombre alumno
en Universidad de Burgos — 31 de mayo de 2020
Tutor: nombre tutor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este punto pasamos a detallar la planificación del proyecto y los pasos que se han dado para poder completarlo con éxito.

A.2. Planificación temporal

La planificación y el desarrollo de este proyecto se han llevado a cabo siguiendo una metodología ágil, concretamente scrum, teniendo que en cuenta que la plataforma de desarrollo usada, GitLab, presenta una nomenclatura propia que no siempre coincide con la de scrum. Así, por ejemplo, GitLab usa *milestones* o *issues* para referirse a *sprints* o *product backlog items (PBIs)*. En este documento intentaremos usar los terminos de scrum siempre que sea posible.

En relación a la plataforma, otro punto a reseñar es la no disponibilidad de gráficas de *burndown* en la versión gratuita. Se ha intentado usar otras alternativas, como Screenful, pero requiere acceso completo a la API de GitLab. Estando el proyecto alojado en la cuenta privada de la empresa HP SCDS, esta no es una opción viable. Así pues, la realización del proyecto se ha llevado a cabo sin disponer de estos gráficos.

El proyecto se ha llevado a cabo siguiendo un sprint de dos semanas, al final de las cuales, en reunión con los tutores, tanto por parte de la Universidad de Burgos como por parte de HP SCDS, se mostraba el trabajo realizado, se proponian cambios o mejoras, y se establecian los objetivos

para el siguiente sprint. El seguimiento de los PBIs se ha realizado a través del tablero Kanban que GitLab provee por defecto.

Pasamos a ver con más detalle el trabajo realizado en cada uno de los sprints:

01 - Arranque del proyecto [10/02/2020 -– 17/02/2020]

El sprint de arranque del proyecto es fundamental teórico, dedicado a estudiar y decidir como vamos a hacer el proyecto, con que herramientas y que medidas usaremos para determinar la validez de un modelo. Este sprint inicial solamente duró una semana, para de esta forma, trás la primera reunión con los tutores el 17 de febrero, poder comenzar un nuevo sprint.

- Estudio comparativo de librerías de machine learning: a la hora de abordar la realización del proyecto nos encontramos con multitud de librerías disponibles. Como primera tarea seleccionamos varias de ellas como posibles candidatas y procedemos a estudiar sus características distintivas.

- Análisis estadístico: en esta tarea nos dedicamos a estudiar métricas alternativas a la precisión para valorar nuestro modelo, como la sensibilidad y la especificidad, así las curvas ROC *receiver oOperating characteristic, o característica operativa del receptor*.

- Estudio documentación de un proceso experimental con CRISP_DM: aquí aprendemos una metodología habitúal a la hora de trabajar en ciencias de datos que usaremos durante el proyecto.

- Elección de librerias: una vez disponemos de los datos necesarios, valoramos como cada librería se adapta a las necesidades del proyecto y seleccionamos una, Pytorch.

- Estudio teórico del perceptrón multicapa: como última tarea del sprint, estudiamos el modelo básico de red neuronal profunda, el perceptrón multicapa.

02 - Desarrollo del perceptrón [02/03/2020 – 16/03/2020]

En este sprint nos enfocamos en la parte formal del TFG, estudiando LaTex y las plantillas de la memoria así como la estructura de archivos, y en los primeros pasos con Pytorch y el dataset.

- Estudio de LaTex y plantillas de memoria: estudiamos los documentos a entregar y sus diferentes secciones a la vez que preparamos el entorno para trabajar con documentos LaTex.
- Añadir a la memoria LaTex la documentación generada en makdown: pasamos la documentación generada en el primer sprint, escrita en markdown en la wiki del proyecto, a LaTex y la incorporamos a la memoria.
- Estructurar las carpetas del repositorio y actualizar fichero Readme.md: procedemos a estructurar los archivos de nuestro solución siguiendo ejemplos de proyectos de exito y otros TFG.
- Aprendizaje de Pytorch: en este punto nos centramos en aprender la librería que vamos a usar.
- Preprocesado de los datos: finalmente estudiamos los datos con los que vamos a trabajar y realizamos una implementación básica de red.

03 - Modelo de perceptrón [16/03/2020 - 06/04/2020]

Por circunstancias personales imprevistas, este sprint se alarga una semana más. En este tiempo apenas se puede avanzar en el proyecto; solo se completa una tarea y se comienza a trabajar en el modelo básico del perceptrón aunque no se completa.

- Gestión de la configuración del repositorio GitLab, donde limpiamos el repositorio de ficheros temporales de Python y Pytorch

04 - Finalizar perceptrón [07/04/2020 - 20/04/2020]

En este sprint nos centramos en el desarrollo de los diversos modelos del perceptrón.

- Modelo simple de perceptron: terminamos la tarea comenzada en el sprint anterior donde generamos nuestro modelo base de percetrón probando con los algoritmos SGD y Adam.
- Entrenar perceptrón con BCEWithLogitsLoss: ampliamos nuestro modelo inicial para usar la función de perdida BCEWithLogitsLoss.
- Mejorar modelo de perceptrón: terminado nuestro modelo base, pasamos ahora a incluir el filtro gaussiano.
- Añadir SMOTE en el modelo de perceptrón: procedemos a entrenar nuevos modelos de percetrón usando SMOTE.
- Hacer análisis de frecuencia con la transformada de Fourier: para terminar con los modelos basados en perceptrón, cambiamos el análisis de la intensidad de luz por el de la frecuencia.
- Caso de estudio de documentación TFG en LaTex relacionado: comprobamos otros TFG para estudiar diferentes enfoques a la hora de encarar la escritura de la memoria.
- Crear notebook para testear los modelos: para finalizar el sprint, creamos un notebook donde poder ejecutar nuestros modelos contra el dataset de testing.

05 - Probar LSTM y crear web [21/04/2020 - 04/05/2020]

En este sprint nos marcamos como objetivo desarrollar modelos basados en redes LSTM y crear una primera versión de aplicación web para poder ejecutar los modelos.

- Formación en conjuntos de datos desequilibrados: estudiamos algunos métodos adicionales para trabajar con dataset desbalanceados.
- Integración documentación Wiki a memoria LateX: migramos el resto de información que teniamos en la wiki a la memoria.
- Crear modelo de red LSTM: desarrollmos varios modelos usando redes LSTMs.
- Crear web para testear modelos: creamos una primera versión donde sea posible subir un archivo para probar la ejecución del modelo.

- Escribir capítulos de Introducción y Objetivos de la memoria: pasamos a completar los capítulos iniciales de la memoria.

06 - Preparar memoria [05/05/2020 - 18/05/2020]

Estaba previsto que el foco de este sprint fuese trabajar en la memoria pero algunos problemas en las tareas relacionadas con la aplicación web provocan que no haya mucho avance en la documentación.

- Desplegar web en Heroku: desplegamos de forma manual la web en Heroku para comprobar que todo funciona correctamente.
- Crear pipeline de release: después del despliegue manual procedemos a automatizar el proceso generando un pipeline en GitLab.
- Completar web: añadimos algunas páginas de información sobre la misión Kepler, sobre los datos y sobre el proyecto. También la localizamos soportando el idioma inglés.
- Ampliar descripcion de las herramientas usadas: añadimos más detalles sobre las herramientas usadas en el proyecto.

07 - Anexos [19/05/2020 – 31/05/2020]

Entrando ya en las etapas finales, este sprint se dedica a continuar el trabajo en la memoria y a ir puliendo detalles del proyecto en general.

- Documentar proceso experimental: completamos el apartado quinta de la memoria detallando las lecciones aprendidas durante el proyecto.
- Incluir dataset y datos del modelo en la web: ampliamos la web incluyendo un dataset de pruebas, información sobre el modelo desplegado y enlazando al repositorio original de los datos en Kaggle.
- Evaluar la calidad del código: usamos una herramienta para controlar la calidad de nuestro código, Codebeat.
- Añadir badges al repositorio: incluimos algunos badges en el repositorio y en el Readme.md para conocer la calidad del código, como fue la ejecución del pipeline de CD o como se encuentra la web en Heroku.

A.3. Estudio de viabilidad

Abordamos a continuación un breve estudio de la viabilidad tanto legal como económica de este proyecto.

Viabilidad económica

Este proyecto nunca se ha enfocado a una viabilidad económica y sería difícil venderlo *tal cual*. Quizá la razón más importante es que apenas hay demanda; el conocimiento de que una estrella es orbitada o no por exoplanetas es, a día de hoy, de poca utilidad comercial, por lo que hay pocas, por no decir ninguna, empresas dispuestas a pagar por ello.

A parte de las empresas privadas, tenemos a diferentes organismos públicos interesados en la exploración espacial, como la NASA, la ESA, la CNSA o similares. Estas agencias se enfrentan a un grave problema común: hay muchos estrellas con posibles exoplanetas y disponen de pocos recursos para explorarlos todos. Dado el carácter probabilístico de este tipo de modelos, puede tener sentido comercial disponer de diferentes aplicaciones y enfocar sus recursos en estudiar aquellas estrellas seleccionadas como más probables por todos, o la mayoría, de las aplicaciones de las que dispongan. A nivel de modelos, en aprendizaje automático, tenemos los metodos de *ensemble* que, básicamente, consisten en combinar varios modelos base para producir otro modelo óptimo. Así pues, no sería ilógico realizar un proceso similar a nivel de aplicaciones.

Por otro lado, otro colectivo que podría estar interesado en nuestra aplicación sería la comunidad científica, más alla de aquellos que trabajan en las agencias espaciales. La confirmación de la presencia o no de exoplanetas en torno a diferentes estrellas puede ayudar a generar nuevo conocimiento sobre este tipo de sistemas, sirviendo de base para confirmar hipótesis o para descartar teorías erróneas.

Finalmente, tenemos otro colectivo que podría estar interesado en nuestro proyecto, los aficionados a la astronomía. En este caso, se podría complementar la aplicación con alguna funcionalidad más como, por ejemplo, proporcionar las coordenadas de la estrella en la que se han identificado exoplanetas de forma que el aficionado pueda apuntar su telescopio diréctamente a ella.

Viabilidad legal

En lo referente a la viabilidad legal del proyecto debemos separarla en dos puntos: las herramientas y los datos. Respecto a las primeras, no

se encuentra ningún problema. Todas las herramientas pueden usarse en aplicaciones comerciales, presentando licencias permisivas (BSD o MIT en su mayoría).

En el apartado de los datos si podemos encontrarnos con mayor problema. Los usados para el entrenamiento de estos modelos, pertenecientes a la campaña 3, son de dominio público, ofrecidos por el Mikulski Archive [1], esperando sólamente ser citados por investigadores. Sin embargo, si deseamos mejorar nuestros modelos con datos más recientes de otras campañas si podemos encontrarnos con datos restringidos o con copyright de forma temporal. Además, en el caso de querer implementar la versión para aficionados a la astronomía, el catálogo de estrellas si tiene copyright.

Apéndice B

Especificación de Requisitos

B.1. Introducción

A continuación, procedemos a abordar los requisitos del proyecto.

B.2. Objetivos generales

Este proyecto de investigación nace con los siguientes objetivos:

- Investigar las técnicas y soluciones existentes de aprendizaje automático para la detección de exoplanetas mediante el análisis del flujo de luz.
- Estudiar los datos disponibles, las diferentes formas de procesarlos y como este procesado puede ayudarnos a obtener mejores resultados.
- Diseñar, implementar y comparar diferentes soluciones, seleccionando el modelo que presente mejores resultados.
- Disponer de software que facilite el uso de diferentes técnicas y el desarrollo de modelos alternativos a los aquí presentados.

Además, durante el desarrollo del proyecto se considera añadir algunos nuevos requisitos de importancia secundaria:

- La creación de una interfaz gráfica que permita ejecutar el modelo para comprobar los datos contenidos en un archivo.

- Desarrollar un pipeline de despliegue continuo que permita desplegar de forma automatica cualquier cambio en nuestro modelo o en la interfaz grafica.

B.3. Catalogo de requisitos

Se muestran a continuación las características que debe cumplir nuestro proyecto, clasificándolas en dos grupos, requisitos funcionales y no funcionales. Mientras que los primeros definen un comportamiento exacto del software, es decir, *que* hace, los segundos hacen referencia a sus propiedades, es decir, *como* las hace.

Definimos dos actores:

- **Investigador**: persona que realiza la investigación y utiliza el software para generar y entrenar modelos.
- **Usuario**: persona que usa el software para determinar si un conjunto de estrellas tienen o no exoplanetas orbitando en torno a ellas.

Requisitos funcionales

- **RF-1** El investigador debe poder dividir un conjunto de datos en sets de entrenamiento y validación.
  - **RF-1.1** Se debe indicar que porcentaje respecto al total de instancias se incluiran en el set de validación.
  - **RF-1.2** La inclusión de una instancia concreta en un set u otro debe ser aleatoria pero se podrá indicar una semilla para que, dado un set de entrada concreto y un porcentaje determinado, la división produzca los mismos set de entrenamiento y validación.
- **RF-2** El investigador debe poder aplicar técnicas y algoritmos que prepararen los datos.
  - **RF-2.1** Debe poder normalizar los datos.
  - **RF-2.2** Debe poder aplicar un filtro gaussiano.
  - **RF-2.3** Debe poder eliminar los picos anormales de intensidad de luz.

    - **RF-2.4** Debe poder transformar el conjunto de datos desde el dominio de la intensidad al dominio de la frecuencia.

- **RF-3** El investigador debe poder mostrar gráficas de la frecuencia o intensidad de la luz respecto al tiempo.

- **RF-4** El investigador debe poder instanciar modelos de redes neuronales.

    - **RF-4.1** Debe poder instanciar modelos de perceptrón multicapa configurables.
    - **RF-4.2** Debe poder instanciar modelos de red LSTM configurable.

- **RF-5** El investigador debe poder entrenar un modelo.

    - **RF-5.1** El investigador debe poder elegir diferentes algoritmos de optimización o funciones de coste.
    - **RF-5.2** Debe mostrarse la evolución del entrenamiento.
    - **RF-5.3** Debe guardarse el mejor modelo generado durante el entrenamiento según el nombre y la ruta indicada por el investigador.
    - **RF-5.4** El investigador debe poder mostrar gráficamente los resultados del entrenamiento.

- **RF-6** El investigador debe poder probar un modelo guardado.

    - **RF-6.1** El investigador debe poder indicar el nombre y la ruta del modelo a cargar.

- **RF-7** El usuario debe poder analizar un archivo con datos en la aplicación.

    - **RF-7.1** El usuario debe tener información suficiente sobre el formato del archivo a procesar.
    - **RF-7.2** El usuario debe obtener que estrellas presentan exoplanetas.
    - **RF-7.3** La aplicación debe mostrar las gráficas correspondientes a las estrellas con exoplanetas.

Requisitos no funcionales

- **RNF-1** El entrenamiento o el testeo de modelos debe poder llevarse a cabo tanto en CPUs como en GPUs, según el hardware del que el investigador disponga.
- **RNF-2** Debe proveerse una aplicación para que un usuario sin entorno ni conocimientos de programación puede probar alguno de los modelos entrenados.

B.4. Especificación de requisitos

Diagrama de casos de uso del investigador

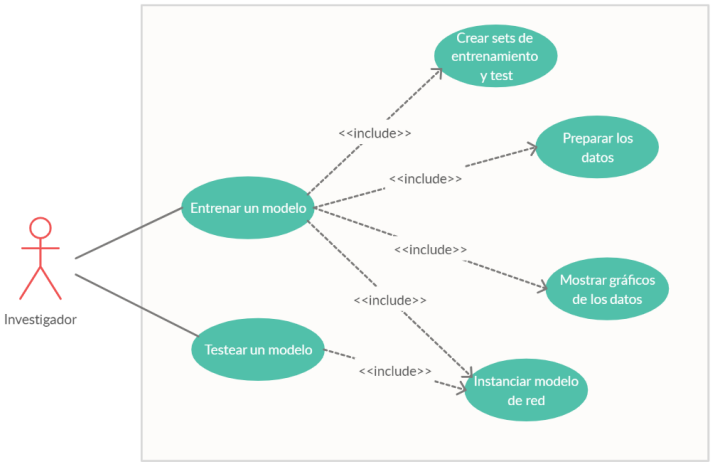

Figura B.1: Diagrama de casos de uso del investigador

Diagrama de casos de uso del usuario

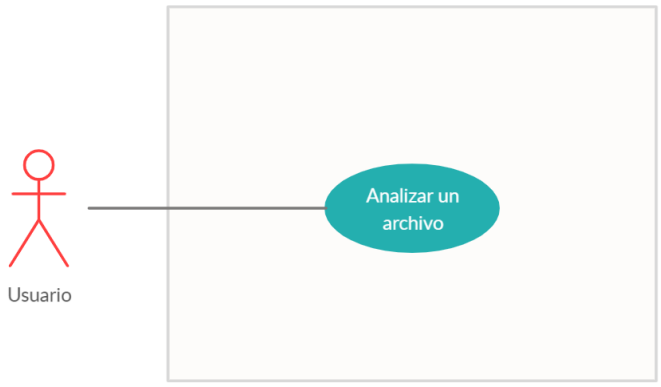

Figura B.2: Diagrama de casos de uso del usuario

Especificaciones de los casos de uso

Caso de uso	Entrenar un modelo
Requisitos	RF-5 RF-5.1 RF-5.2 RF-5.3 RF-5.4
Descripción	El investigador puede entrenar un nuevo modelo dado un conjunto de datos usando un algoritmo de optimización y una función de coste.
Precondiciones	El investigador debe disponer de un conjunto de datos.
Acciones	1. El investigador carga los datos. 2. Opcionalmente, el investigador puede dividir sus datos en dos conjuntos distintos, entrenamiento y validación. 3. Opcionalmente, el investigador puede preparar los datos usando alguna de las técnicas y herramientas propuestas. 4. Opcionalmente, el investigador visualizar los datos. 5. El investigador elige un tipo de modelo y genera una instancia. 6. El investigador fija los hiperparámetros según deseé. 7. El investigador decide que algoritmo de optimización y función de coste usar. 8. El investigador comienza el entrenamiento. 9. Opcionalmente, cuando el entrenamiento termine, el investigador puede mostrar gráficamente los resultados del mismo.
Postcondiciones	Se ha generado un modelo y las imagenes con los resultados del entrenamiento.
Excepciones	Formato de datos incorrecto. Algoritmo de optimización invalido. Función de coste invalida.Configuración de la instancia del modelo incorrecta.
Importancia	Alta

Tabla B.1: Caso de uso ”Entrenar un modelo”.

Caso de uso	Crear sets de entrenamiento y validación
Requisitos	RF-1 RF-1.1 RF-1.2
Descripción	A partir de un conjunto de datos, el investigador puede dividirlo en dos de forma aleatoria y segun la proporción deseada.
Precondiciones	El investigador debe disponer de un conjunto de datos.
Acciones	1. El investigador selecciona el conjunto de datos. 2. El investigador elige que porcentaje del total ocupará el set de validación 3. Opcionalmente, el investigador elige una semilla para la selección aleatoria.
Postcondiciones	El investigador obtiene los datos divididos en dos conjuntos.
Excepciones	El conjunto de datos está vacio. La proporción es uno o mayor.
Importancia	Media.

Tabla B.2: Caso de uso "Crear sets de entrenamiento y validación".

Caso de uso	Preparar los datos
Requisitos	RF-2 RF-2.1 RF-2.2 RF-2.3 RF-2.4
Descripción	El investigador debe poder aplicar a los datos las funciones usadas durante el entrenamiento: normalización, eliminación de picos de intensidad, filtro gaussiano y transformada de Fourier.
Precondiciones	El investigador debe disponer de un conjunto de datos.
Acciones	1. El investigador selecciona el conjunto de datos. 2. El investigador aplica la función correspondiente.
Postcondiciones	El conjunto de datos transformado.
Excepciones	El conjunto de datos está vacio.
Importancia	Media.

Tabla B.3: Caso de uso "Preparar los datos".

Caso de uso	Mostrar gráficos de los datos
Requisitos	RF-3
Descripción	El investigador puede mostrar la gráfica de cualquiera de las estrellas incluidas en el conjunto de datos.
Precondiciones	El investigador debe disponer de un conjunto de datos.
Acciones	1. El investigador selecciona el conjunto de datos. 2. El investigador indica los indices de las estrellas a mostrar. 3. El investigador aplica la función.
Postcondiciones	Gráfica(s) mostrando los datos correspondientes.
Excepciones	El indice de las estrellas seleccionadas no se encuentra en el conjunto de datos.
Importancia	Baja.

Tabla B.4: Caso de uso "Mostrar gráficos de los datos".

Caso de uso	Instanciar modelo de red
Requisitos	RF-4 RF-4.1 RF-4.2
Descripción	El investigador debe poder instanciar modelos de perceptrón o de LSTM.
Precondiciones	Ninguna.
Acciones	1. El investigador selecciona el tipo de modelo a instanciar. 2. Opcionalmente, el investigador los parámetros para el modelo.
Postcondiciones	Instancia del modelo.
Excepciones	Ninguna.
Importancia	Alta.

Tabla B.5: Caso de uso "Instanciar modelo de red".

Caso de uso	Testear un modelo
Requisitos	RF-6 RF-6.1
Descripción	El investigador debe poder hacer predicciones sobre un conjunto de datos con un modelo previamente entrenado.
Precondiciones	El investigador debe contar un modelo entrenado guardado en disco.
Acciones	1. El investigador selecciona el conjunto de datos. 2. El investigador instancia un modelo de red acorde al modelo que desee testear. 3. El investigador testea el modelo.
Postcondiciones	Matriz de confusión del modelo.
Excepciones	El modelo de red instanciado no es el adecuado para el modelo guardado.
Importancia	Alta.

Tabla B.6: Caso de uso "Testear un modelo".

Caso de uso	Analizar un archivo
Requisitos	RF-7 RF-7.1 RF-7.2 RF-7.3
Descripción	El usuario debe poder cargar un archivo en la aplicación, obteniendo predicciones sobre que estrellas presentan exoplanetas así como las gráficas de dichas estrellas.
Precondiciones	El usuario debe disponer de un archivo con datos.
Acciones	1. El usuario selecciona el archivo de datos. 2. El usuario carga el archivo en la aplicación.
Postcondiciones	Resultado del análisis con las gráficas correspondientes.
Excepciones	El archivo no es válido.
Importancia	Media.

Tabla B.7: Caso de uso "Analizar un archivo".

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario

Bibliografía

[1] K2 mission. https://archive.stsci.edu/k2/. Accessed: 21/05/2020.