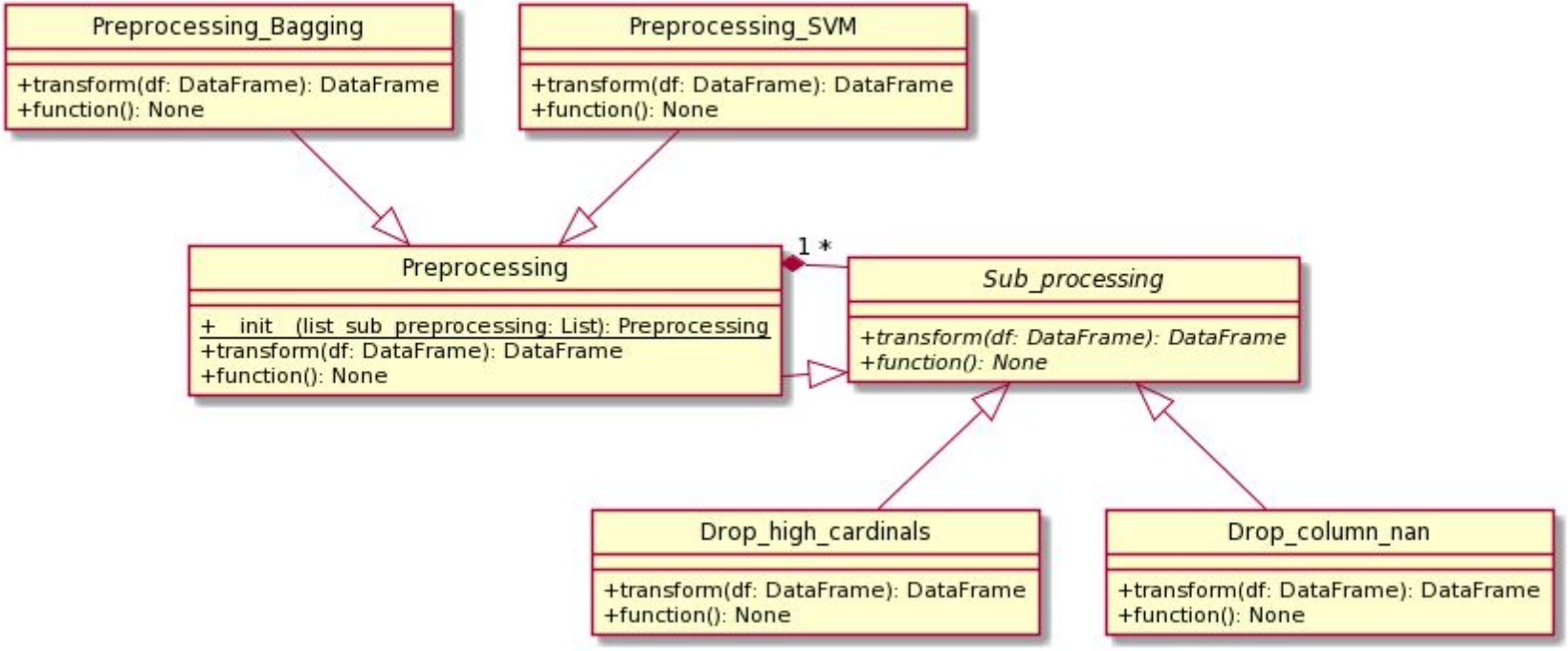
Tablas

Para los pre-procesamientos se siguió un patrón de diseño llamado composite, se dividió todos los pre-procesamientos en tareas más pequeñas llamadas sub preprocessing y luego se fueron uniendo diferentes sub preprocessing formando así pre-procesamientos más complejos.

El diagrama con algunas clases quedaría así:



1. **Tabla de sub preprocesamientos:**

Nombre del sub preprocesamiento	Explicación simple del funcionamiento	Nombre de la clase de python
Drop high cardinals	Elimina las columnas con alta cardinalidad del df	Drop_high_cardinals
Dummy variables	Transforma las columnas categóricas en dummy variables	Dummy_variables
Drop column nan	Elimina la columna con muchos nan (mayor al 70%)	Drop_column_nan
Nan to zero	Rellena los nan de edad con 0	Nan_to_zero
Nan to mean	Rellena los nan de edad con la media	Nan_to_mean
Nan to mode	Rellena los nan de edad con la moda	Nan_to_mode
Nan to median	Rellena los nan de edad con la mediana	Nan_to_median
Standardizer columns	Estandariza los atributos no binarios	Std_columns
Normalizer columns	Normaliza los atributos no binarios	Normalizer_columns

2. Tabla de preprocesamientos:

Nombre del preprocesamiento	Explicación simple del funcionamiento	Nombre de la clase de python
Preprocessing Dessicion Tree Classifier	Elimina las columnas con alta cardinalidad. Elimina las columnas con un porcentaje de valores nan mayor igual al 70%. Transforma las variables categóricas en dummy variables. Rellena los nan de edad con la mediana.	Preprocessing_Tree
Preprocessing K Nearest Neighbors	Elimina las columnas con alta cardinalidad del df. Transforma las columnas categóricas en dummy variables. Rellena los nan de edad con la mediana. Estandariza los atributos no binarios.	Preprocessing_KNN
Preprocessing Naive Bayes, Random Forest and Stacking Classifier	Elimina las columnas con alta cardinalidad. Elimina las columnas con un porcentaje de valores nan mayor igual al 70%. Transforma las variables categóricas en dummy variables. Rellena los nan de edad con la mediana.	Preprocessing_NB_RF_SC

Preprocessing Support Vector Machine	Elimina las columnas con alta cardinalidad del df. Elimina las columnas con un porcentaje de valores nan mayor al 70%. Transforma las columnas categóricas en dummy variables. Rellena los nan de edad con la moda. Normaliza los atributos no binarios.	Preprocessing_SVM
Preprocessing Bagging and Boosting	Elimina las columnas con alta cardinalidad. Elimina las columnas con un porcentaje de valores nan mayor igual al 70%. Transforma las variables categóricas en dummy variables. Rellena los nan de edad con la moda.	Preprocessing_Bagging_Boosting

3. Tabla de los modelos:

En el caso de Neural Network Classifier no se pudo asegurar la reproducibilidad, por eso no posee un preprocesamiento dado que este varía cada vez que corre el notebook, las métricas son las últimas obtenidas.

Nombre del modelo	Nombre del pre procesamiento	AUC-ROC	Accuracy	Precision	Recall	F1
Dessicion Tree Classifier	Preprocessing_Tree	0.838710	0.790123	0.791667	0.612903	0.690909
KNN Classifier	Preprocessing_KNN	0.859355	0.827160	0.869565	0.645161	0.740741
Naive Bayes Classifier	Preprocessing_NB_RF_SC	0.785484	0.716049	0.681818	0.483871	0.566038
Support Vector Machine Classifier	Preprocessing_SVM	0.866129	0.839506	0.909091	0.645161	0.754717
Neural Network Classifier	-	0.837097	0.839506	0.800000	0.774194	0.786885
Bagging Classifier	Preprocessing_Bagging_Boosting	0.871613	0.851852	0.913043	0.677419	0.777778
Random Forest Classifier	Preprocessing_NB_RF_SC	0.852258	0.814815	0.863636	0.612903	0.716981
Boosting Classifier	Preprocessing_Bagging_Boosting	0.867097	0.839506	0.846154	0.709677	0.771930
Stacking Classifier	Preprocessing_NB_RF_SC	0.862258	0.827160	0.840000	0.677419	0.750000

4. Conclusiones:

Al hacer este TP se pudo observar que todos los modelos obtuvieron un **Recall** bajo en comparación al resto de las métricas, esto se debe a que el modelo fue escogido a base de la métrica auc roc score y esta métrica busca tener un alto **False Positive Rate** y **True Positive Rate**, donde este último es la **Precision** el cual tiene una relación inversa con el **Recall**.

Al final los mejores modelos resultaron ser:

1. Bagging Classifier
2. Boosting Classifier
3. Support Vector Machine Classifier

El mejor modelo parece ser el **Bagging Classifier** dado que no solo tiene el mejor **AUC-ROC** score, sino que también tiene el resto de las métricas con un valor alto, sobretodo el **Precision** que es de **0.913043** . Por eso recomendaría el **Bagging Classifier.**

Si comparamos este modelo con el baseline hecho en la parte uno; además de que la métrica no es confiable dado que no se hizo una separación del dataset original por lo cual se entrenó y comparó con los mismos datos; dado que este fue hecho a mano es muy posible que tenga un sesgo muy alto debido a mis presupuestos basados en el análisis de los datos a ojo, en cambio este modelo final fue hecho es un ensamble con el cual está demostrado que es mejor a cualquier modelo por sí solo además de tener una base matemática que lo sustenta en el estimador base que es un **Decision Tree Classifier**.