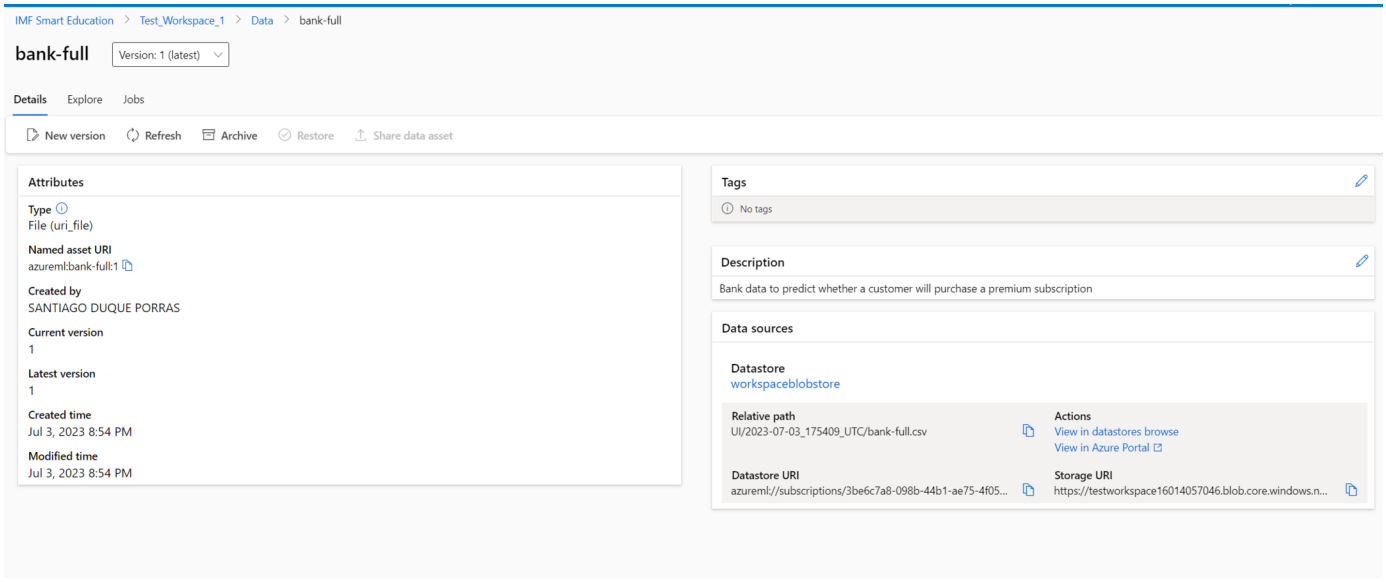
Este documento detalla el desarrollo del modelo ML y la ingeniería de características.

Primero, se realizaron trabajos de preprocesamiento: selección de variables, encoding, normalización, etc.

Se sube el dataset a Azure:



Se prueba a entrenar con diferentes algoritmos, y se adjuntan las métricas de desempeño.

Gradient Boosting:

```
input data: /mnt/azureml/cr/j/36360f3ef50c40ad8c0ffc8cc7
Training with data of shape (38429, 38)
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      6023
           1       0.61      0.20      0.30       759

    accuracy                           0.90      6782
   macro avg       0.76      0.59      0.62      6782
weighted avg       0.87      0.90      0.87      6782

Registering the model via MLFlow
```

Random Forest:

```
input data: /mnt/azureml/cr/j/0fd0bcffa07f4462a0f8400442
Training with data of shape (38429, 38)
              precision    recall  f1-score   support

           0       0.91      0.98      0.95      6021
           1       0.63      0.22      0.33       761

    accuracy                           0.90      6782
   macro avg       0.77      0.60      0.64      6782
weighted avg       0.88      0.90      0.88      6782

Registering the model via MLFlow
```

SVM:

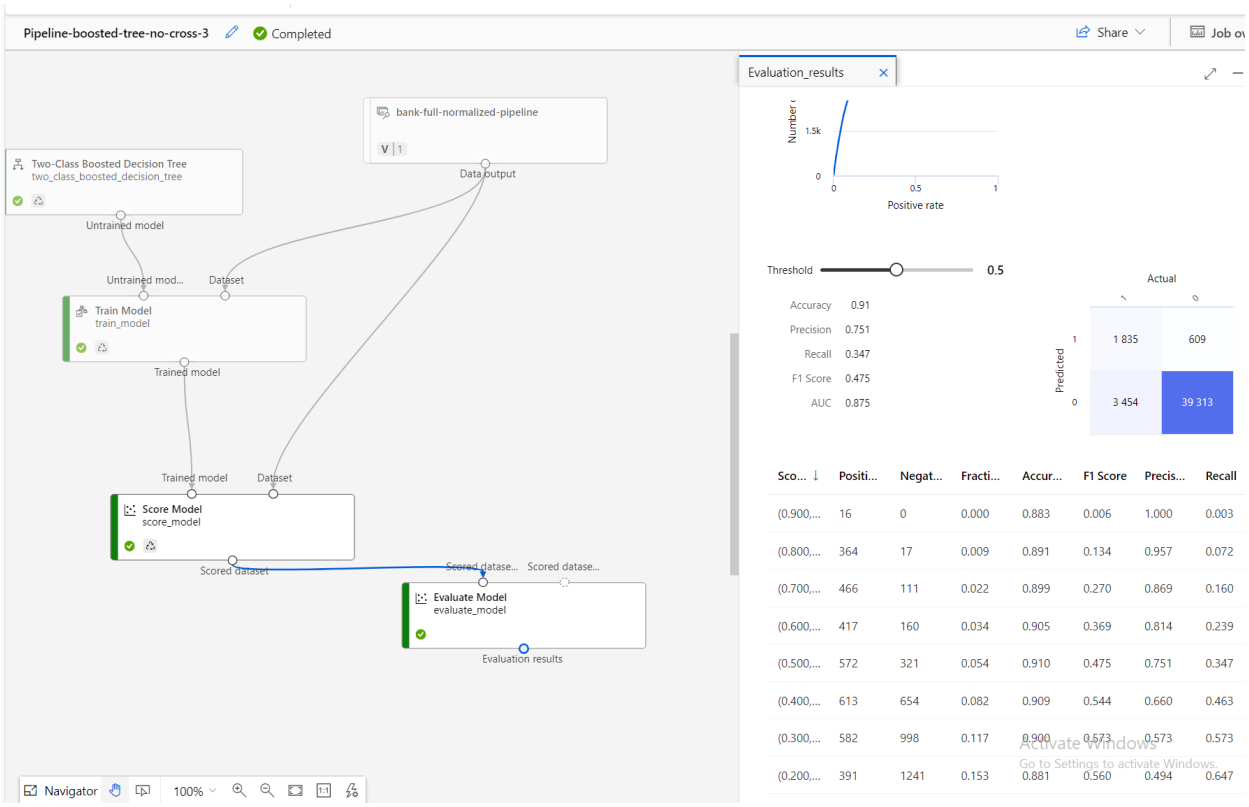
```
[illegible]
Training with data of shape (38429, 38)
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      5970
           1       0.66      0.20      0.31       812

    accuracy                           0.89      6782
   macro avg       0.78      0.59      0.63      6782
weighted avg       0.87      0.89      0.87      6782
```

Perceptron Linear

```
input data: /mnt/azureml/cr/j/0fe1d85b4acb4b3bbdb3f/f4fba
Training with data of shape (38429, 38)
              precision    recall  f1-score   support

           0       0.92      0.76      0.83      6005
           1       0.20      0.46      0.28       777

    accuracy                           0.72      6782
   macro avg       0.56      0.61      0.55      6782
weighted avg       0.83      0.72      0.77      6782

Registering the model via MLFlow
```

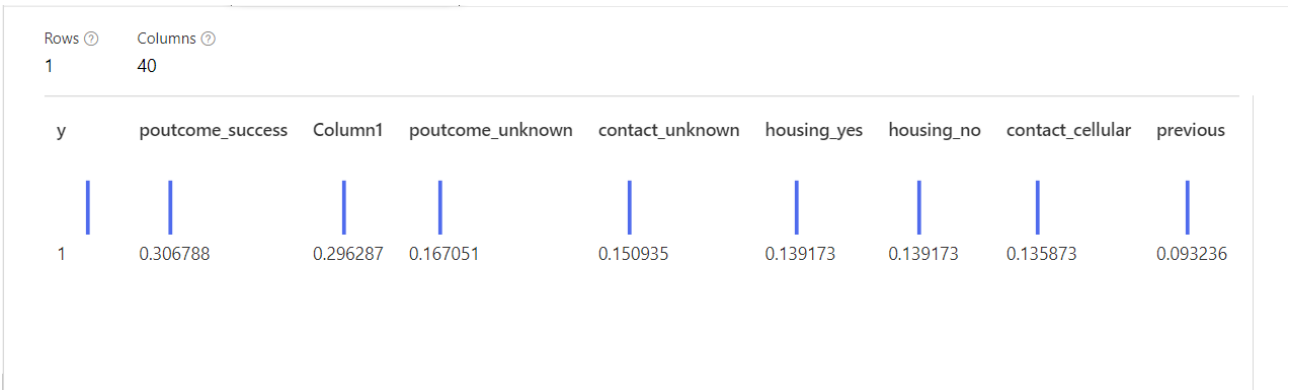
Se prueba también con una variante, Boosted Decision Tree, usando las pipelines de Azure.

Pipeline y resultados:



Se decide continuar el desarrollo con el algoritmo de Random Forest.

Se lleva a cabo un análisis del poder decisorio de variables (PCA):



Se lleva a cabo un ajuste de los hiperparámetros del algoritmo Random Forest usando una grid search:

```
# Hyperparameter tuning for the Random Forest Classifier using random grid

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}



#Random Forest
rfc = RandomForestClassifier()
#rfc.fit(X_train, y_train)
#model = rfc

# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rfc, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=42, n_jobs = -1)
# Fit the random search model
rf_random.fit(X_train, y_train)
model = rf_random

print("Grid search best paremeters:")
print(rf_random.best_params_)
rf_random.best_params_
```

Las cinco mejores combinaciones de hiperparámetros encontradas:

```
tuning_results.head()
```

param_n_estimators	param_min_samples_split	param_min_samples_leaf	param_max_features	param_max_depth	param_bootstrap	
200	5	4	auto	10.0	True	'm
1000	5	2	sqrt	10.0	True	'm
1600	5	1	auto	10.0	True	'm
800	10	4	sqrt	50.0	True	'm
1000	2	1	auto	10.0	False	'm

Hay sustancialmente más registros de una categoría que de otra, y esto afecta a la precisión al detectar sobre la categoría objetivo 1 (‘sí’). Para contrarrestarlo, se introduce oversampling de esa categoría en el dataset. Estos son los resultados de desempeño del algoritmo Tuned Random Forest con hiperparámetros optimizados y oversampling:

```
parser=ArgumentParser(prog='main.py', usage=None, descriptio
input data: /mnt/azureml/cr/j/7a0809bf2d524eefb1aed336d9587b
Training with data of shape (40720, 38)
              precision    recall  f1-score   support

           0       0.88      0.98      0.93      6028
           1       0.76      0.32      0.45      1158

    accuracy                           0.87      7186
   macro avg       0.82      0.65      0.69      7186
weighted avg       0.86      0.87      0.85      7186

Registering the model via MLFlow

```

20%

```
parser=ArgumentParser(prog='main.py', usage=None, descri
input data: /mnt/azureml/cr/j/1b339697cdd242a69bab9ee01d
Training with data of shape (42416, 38)
              precision    recall  f1-score   support

           0       0.87      0.97      0.92      5994
           1       0.80      0.44      0.57      1492

    accuracy                           0.87      7486
   macro avg       0.84      0.71      0.74      7486
weighted avg       0.86      0.87      0.85      7486

Registering the model via MLFlow

```

25%

```
2023/07/09 14:50:55 WARNING mlflow.sklearn: Failed to log evaluati
Registered model 'bank-marketing-model-tuned-random-forest' alread
2023/07/09 14:51:00 INFO mlflow.tracking._model_registry.client: W
Created version '9' of model 'bank-marketing-model-tuned-random-fo
parser=ArgumentParser(prog='main.py', usage=None, description=None
input data: /mnt/azureml/cr/j/fecc758a98b84c6cb9c24595386c7cc7/cap
Training with data of shape (44113, 38)
              precision    recall  f1-score   support

           0       0.87      0.96      0.91      6004
           1       0.81      0.51      0.63      1781

    accuracy                           0.86      7785
   macro avg       0.84      0.74      0.77      7785
weighted avg       0.86      0.86      0.85      7785

Registering the model via MLFlow

```

30%

Se ha elegido, por tanto, desarrollar un modelo de Tuned Random Forest con optimización de hiperparámetros y oversampling, mediante el editor de código y el editor de *pipelines* de Azure.

A continuación se muestran los errores tras intentar desplegar el modelo (al ser una cuenta gratuita carece de suficiente memoria), primero el error en el editor de código y segundo la pantalla de error de deployment

```
# create the online deployment
blue_deployment = ml_client.online_deployments.begin_create_or_update(
    blue_deployment
).result()

# blue deployment takes 100% traffic
# expect the deployment to take approximately 8 to 10 minutes.
endpoint.traffic = {"blue": 100}
ml_client.online_endpoints.begin_create_or_update(endpoint).result()
```

[10] 1 sec HttpResponseError: (BadRequest) The request is invalid. Code: BadRequest Message: The request is invalid. Exception Details: (InferencingClientCreateDeploymentFailed) InferencingClient HttpRequest error, error detail: {"errors":{"VmSize":["Not enough quota available for Stan

```
Check: endpoint bank-endpoint-66afe121 exists
data_collector is not a known attribute of class <class 'azure.ai.ml._restclient.v2022_02_01_preview.models._models_py3.ManagedOnlineDeployment'> and will be ignored
---------------------------------------------------------------------------
HttpResponseError                         Traceback (most recent call last)
Cell In[10], line 2
      1 # create the online deployment
----> 2 blue_deployment = ml_client.online_deployments.begin_create_or_update(
      3     blue_deployment
      4 ).result()
      6 # blue deployment takes 100% traffic
      7 # expect the deployment to take approximately 8 to 10 minutes.
      8 endpoint.traffic = {"blue": 100}

File /anaconda/envs/azureml_py310_sdkv2/lib/python3.10/site-packages/azure/core/tracing/decorator.py:76, in distributed_trace.<locals>.decorator.<locals>.wrapper_use_tracer(*args, **kwargs)
     74 span_impl_type = settings.tracing_implementation()
```

Live traffic allocation

bank-marketing-model-tuned-ran-4 (0%)

Mirrored traffic allocation

--

Deployment bank-marketing-model-tuned-ran-4

Name
bank-marketing-model-tuned-ran-4

Live traffic
0%

Scoring script
Auto-generated

Provisioning state
Failed

Error details

ResourceOperationFailure: OutOfQuota: Container terminated due to insufficient memory. Please see troubleshooting guide, available here: https://aka.ms/oe-tsg...

SKU
Standard_DS1_v2

Egress public network access
Enabled

Instance count
1

Scaling
Configure auto scaling

Model ID
bank-marketing-model-tuned-random-forest:4

Environment
Auto-generated

Application Insights enabled
false

Logs
Deployment logs

Metrics