PROJECTE MINERIA DE DADES

DETECCIÓ DE PHISHING SOBRE PÀGINES WEB

MIDA

24 Octubre 2023

Eric Gonzalez Duro

ÍNDEX

1. Introducció

En un món cada vegada més interconnectat, la seguretat cibernètica s'ha convertit en una preocupació crítica. El phishing, en particular, ha sorgit com una de les formes més efectives en les quals els ciberdelinqüents poden enganyar els usuaris per a obtenir informació personal i financera sensible. Aquests atacs sofisticats i cada vegada més difícils de detectar representen una amenaça constant per a la integritat i la privacitat en línia.

En aquest projecte, s'aborda la tasca de desenvolupar un sistema de detecció de phishing basat en aprenentatge automàtic. S'utilitzarà un conjunt de dades que inclou 11,430 URL amb 87 característiques extretes. Aquestes característiques es divideixen en tres classes: 56 extretes de l'estructura i sintaxi de les URL, 24 extretes del contingut de les pàgines corresponents i 7 extretes mitjançant consultes a serveis externs. El que fa que aquest projecte sigui especialment desafiador és que el conjunt de dades està equilibrat, amb un 50% d'URL de phishing i un 50% d'URL legítimes, la qual cosa reflecteix la realitat de la ciberdelinqüència en línia.

L'objectiu principal d'aquest treball és aplicar tècniques de mineria de dades per a desenvolupar un model capaç de distingir entre URLs legítimes i URLs de phishing amb precisió. Per a aconseguir aquest objectiu, es duran a terme una sèrie de passos crucials que involucren la selecció d'un conjunt de dades no trivial, preprocessament de dades, ajust de paràmetres d'algorismes d'aprenentatge automàtic, interpretació de models i, finalment, una avaluació exhaustiva de diferents enfocaments.

Aquest projecte se centra en comprendre i aplicar el procés complet de mineria de dades, des de la selecció del conjunt de dades fins a la discussió crítica dels resultats obtinguts. La precisió dels models no és l'únic resultat rellevant; la presa de decisions fonamentades i l'avaluació crítica dels resultats exerciran un paper crucial en l'avaluació d'aquest projecte.

En el següent treball, es detallarà el procés complet, des de la selecció del conjunt de dades fins a la discussió de les conclusions finals. S'explicaran els mètodes utilitzats, els enfocaments de preprocessament i ajust de paràmetres, i es presentaran i analitzaran els resultats obtinguts. Aquest projecte té com a objectiu no sols desenvolupar models efectius de detecció de phishing, sinó també brindar una comprensió profunda de les etapes de la mineria de dades i la lògica darrere de cada decisió presa en el procés.

2. Descripció de les dades originals

El dataset que utilitzarem per a poder realitzar aquesta pràctica, s'ha obtingut del següent link:

https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset/

Tot i que a la descripció del dataset de Kaggle, s'esmenta que la font real es del següent link:

https://data.mendeley.com/datasets/c2gw7fy2j4/3

Per altra banda, el conjunt de dades consta de 11,431 files i 89 columnes. Aquestes columnes representen diverses característiques que s'han extret de les URL i s'utilitzen per a la tasca de detecció de phishing. Les dades inclouen una varietat de tipus de dades com números enters, nombres de coma flotant i indicadors binaris. Cada columna té un significat específic i representa una característica diferent que es pot utilitzar per avaluar la probabilitat que una URL sigui de pishing o legítima. A continuació, es proporciona una descripció general d'aquestes columnes, la qual cosa ajudarà a comprendre millor la naturalesa de les dades originals:

url → La URL completa.
length_url → Longitud de la URL.
length_hostname → Longitud del nom d'amfitrió.
ip → Indicador de si la URL conté una adreça IP en lloc d'un nom d'amfitrió.
nb_dots → Número de punts a la URL.
nb_hyphens → Número de guions a la URL.
nb_at → Número de símbols "@" a la URL.
nb_qm → Número de signes d'interrogació a la URL.
nb_and → Número de signes "&" a la URL.
nb_or → Número de signes "|" a la URL.
nb_eq → Número de signes "=" a la URL.
nb_underscore → Número de guions baixos a la URL.
nb_tilde → Número de virgulilles a la URL.
nb_percent → Número de signes de percentatge a la URL.
nb_slash → Número de barres diagonals a la URL.

nb_star → Número d'asteriscs a la URL.

nb_colon → Número de dos punts a la URL.

nb_comma → Número de comes a la URL.

nb_semicolon → Número de punts i comes a la URL.

nb_dollar → Número de signes de dòlar a la URL.

nb_space → Número d'espais a la URL.

nb_www → Indicador de si "www" està present a la URL.

nb_com → Número de vegades que apareix ".com" a la URL.

nb_dslash → Número de barres diagonals dobles a la URL.

http_in_path → Indicador de si "http" apareix a la ruta de la URL.

https_token → Indicador de si es troba el token "https" a la URL.

ratio_digits_url → Relació entre dígits i caràcters a la URL.

ratio_digits_host → Relació entre dígits i caràcters al nom d'amfitrió.

punycode → Indicador de si s'utilitza punycode a la URL.

port → Port a la URL.

tld_in_path → Indicador de si el domini de nivell superior (TLD) apareix a la ruta de la URL.

tld_in_subdomain → Indicador de si el TLD apareix en un subdomini.

abnormal_subdomain → Indicador de si el subdomini és anormal.

nb_subdomains → Número de subdominis a la URL.

prefix_suffix → Indicador de si s'utilitzen prefixos o sufixos a la URL.

random_domain → Indicador de si el domini sembla ser aleatori.

shortening_service → Indicador de si s'utilitza un servei d'acurtament d'URL.

path_extension → Extensió de la ruta a la URL.

nb_redirection → Número de redireccions a la URL.

nb_external_redirection → Número de redireccions externes a la URL.

length_words_raw → Longitud de les paraules a la URL original.

char_repeat → Indicador de repeticions de caràcters a la URL.

shortest_words_raw → Longitud de la paraula més curta a la URL original.

shortest_word_host → Longitud de la paraula més curta al nom d'amfitrió.

shortest_word_path → Longitud de la paraula més curta a la ruta.

longest_words_raw → Longitud de la paraula més llarga a la URL original.

longest_word_host → Longitud de la paraula més llarga al nom d'amfitrió.

longest_word_path → Longitud de la paraula més llarga a la ruta.

avg_words_raw → Mitjana de la longitud de les paraules a la URL original.
avg_word_host → Mitjana de la longitud de les paraules al nom d'amfitrió.
avg_word_path → Mitjana de la longitud de les paraules a la ruta.
phish_hints → Indicador de pistes de phishing a la URL.
domain_in_brand → Indicador de si el domini està en una marca registrada.
brand_in_subdomain → Indicador de si la marca està en un subdomini.
brand_in_path → Indicador de si la marca està a la ruta.
suspecious_tld → Indicador de domini de nivell superior sospitós (TLD).
statistical_report → Indicador de si es proporciona un informe estadístic.
nb_hyperlinks → Número d'hiperenllaços a la pàgina.
ratio_intHyperlinks → Relació d'hiperenllaços interns a la pàgina.
ratio_extHyperlinks → Relació d'hiperenllaços externs a la pàgina.
ratio_nullHyperlinks → Relació d'hiperenllaços nuls a la pàgina.
nb_extCSS → Número de fulls d'estil enllaçats externament.
ratio_intRedirection → Relació de redireccions internes a la pàgina.
ratio_extRedirection → Relació de redireccions externes a la pàgina.
ratio_intErrors → Relació d'errors interns a la pàgina.
ratio_extErrors → Relació d'errors externs a la pàgina.
login_form → Indicador de si es troba un formulari d'inici de sessió a la pàgina.
external_favicon → Indicador de si s'utilitza un favicon extern a la pàgina.
links_in_tags → Número d'enllaços en etiquetes HTML.
submit_email → Indicador de si s'envia correu electrònic en fer clic a un enllaç.
ratio_intMedia → Relació d'elements multimèdia interns a la pàgina.
ratio_extMedia → Relació d'elements multimèdia externs a la pàgina.
sfh → Indicador de si existeix "same origin" a la pàgina.
iframe → Indicador de si s'utilitzen iframes a la pàgina.
popup_window → Indicador de si s'utilitzen finestres emergents a la pàgina.
safe_anchor → Indicador de si els ancoratges són segurs.
onmouseover → Indicador de si s'utilitza "onmouseover" a la pàgina.
right_clic → Indicador de si es permet fer clic dret a la pàgina.
empty_title → Indicador de si el títol de la pàgina està buit.
domain_in_title → Indicador de si el domini està en el títol de la pàgina.
domain_with_copyright → Indicador de si el domini té drets d'autor.
whois_registered_domain → Indicador de si el domini està registrat.

domain_registration_length → Longitud del registre del domini.
domain_age → Antiguitat del domini.
web_traffic → Trànsit web del lloc.
dns_record → Registre DNS del lloc.
google_index → Índex de Google del lloc.
page_rank → Rang de la pàgina del lloc.
status → Indica si la URL es legítima o bé es phishing

Aquestes característiques es divideixen en tres classes, que inclouen característiques relacionades amb l'estructura i sintaxi de les URL, característiques extretes del contingut de les pàgines corresponents i característiques obtingudes mitjançant consultes a serveis externs.

En l'anàlisi de les dades originals, explorarem més a fons aquestes característiques per a comprendre la seva rellevància en la detecció de phishing. A més, es considerarà la distribució de les classes de phishing i legítimes en el conjunt de dades equilibrat, la qual cosa influeix en l'avaluació dels models i la interpretació dels resulta

3. Pre-Processament de les dades

En abordar la tasca de processament del conjunt de dades "dataset_phishing.csv", el primer pas va ser la seva càrrega i exploració. Aquesta etapa inicial no només ens proporciona una visió general de les dades, sinó que també destaca qualsevol anomalia o peculiaritat que podria requerir atenció. La nostra anàlisi va revelar que el conjunt de dades tenia diverses característiques, majoritàriament numèriques, i no presentava valors faltants. La presència de dades completes simplifica molts passos posteriors, ja que no cal abordar l'imputació o la eliminació de valors faltants.

La columna "url", tot i ser informativa des d'una perspectiva referencial, no aportava valor discriminatiu per als algoritmes de machine learning. La seva naturalesa textual i única per cada fila podria introduir soroll innecessari o sobreajustament en models posteriors. Per això, es va decidir eliminar-la, simplificant així la dimensionalitat de les dades.

La selecció de característiques és un pas crític que pot influir directament en la eficàcia i eficiència dels models de machine learning. En aplicar un anàlisi de correlació, vam observar que algunes característiques tenien valors constants, cosa que implica que no varien i, per tant, no aporten informació per distingir entre URL legítimes i de phishing. La seva eliminació va ser una elecció lògica per reduir la complexitat sense comprometre la qualitat de les dades.

La normalització de les dades és fonamental, especialment quan les característiques tenen diferents unitats o escales. Algoritmes que utilitzen càlculs de distància o gradient, com ara SVM, k-NN o xarxes neuronals, poden ser altament sensibles a les escales de les dades. Al normalitzar, assegurem que cada característica té el mateix pes, optimitzant així el rendiment dels algoritmes.

Finalment, l'aplicació de PCA va ser una decisió estratègica per reduir la dimensionalitat tot mantenint la major part de la variabilitat de les dades. En un conjunt de dades amb moltes característiques, pot haver-hi una gran quantitat d'informació redundant. El PCA permet identificar i mantenir només les dimensions més informatives. El fet que 60 components capturesin el 95% de la variabilitat

indica que moltes de les característiques originals estaven correlacionades i que aquesta representació compacta podria ser igualment efectiva per a tasques de modelització.

En conclusió, cada pas del pre-processament va ser meticulosament considerat i executat amb l'objectiu d'optimitzar la qualitat i eficiència de les dades per a l'anàlisi i modelització posteriors. Aquesta preparació acurada és fonamental per assegurar que qualsevol model de machine learning que es desenvolupi posteriorment tingui una base sòlida sobre la qual operar.

Anàlisi Exploratori Inicial:

Abans de qualsevol pre-processament, és essencial entendre la naturalesa del conjunt de dades. Per fer-ho, hem visualitzat les primeres files i hem explorat les estadístiques descriptives.

```python
data.head()
data.describe([x*0.1 for x in range(10)])
```

Un dels passos clau de la fase d'anàlisi exploratòria és identificar possibles problemes, com ara valors nuls o dades duplicades.

```python
data.isnull().sum()
data.duplicated().sum()
```

Transformació de Característiques:

La columna 'status', que indica si una URL és "legitimate" o "phishing", es va transformar en una representació numèrica per facilitar l'anàlisi quantitatiu.

```python
data['status_numeric'] = data['status'].apply(lambda x: 1 if x == 'phishing' else 0)
```

Aquesta transformació ens va permetre calcular la correlació de cada característica amb la variable objectiu, ajudant-nos a comprendre quines característiques podrien ser més rellevants per a la detecció de phishing.

Preprocessament de Dades:

Per assegurar-nos que les dades estiguin en un format adequat per a l'anàlisi, hem codificat les variables categòriques i hem normalitzat les dades.

La normalització és especialment important quan es treballa amb algoritmes que són sensibles a l'escala de les característiques, com ara la regressió logística o les màquines de vectors suport.

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(data.select_dtypes(include=[np.number]))
```

Reducció de Dimensions amb PCA:

L'Anàlisi de Components Principals (PCA) és una tècnica que permet reduir la dimensionalitat del conjunt de dades mentre es conserva la major part de la informació.

```python
from sklearn.decomposition import PCA
pca = PCA()
data_pca = pca.fit_transform(data_normalized)
```

Després d'aplicar PCA, vam observar la variància explicada per cada component i vam determinar que 29 components principals eren suficients per capturar almenys el 95% de la variabilitat total de les dades.

```python
variancia_acumulada = pca.explained_variance_ratio_.cumsum()
components_95_variancia = sum(variancia_acumulada < 0.95) + 1
```

Aquesta reducció de dimensions no només simplifica l'anàlisi, sinó que també pot ajudar a millorar la velocitat i l'eficàcia d'alguns algoritmes de mineria de dades.

4. Criteri d'avaluació dels models de mineria de dades

4. 1 Procediment per al desglossament de dades

Tenint en compte que tenim un conjunt de dades equilibrat amb un 50% d'URL de phishing i un 50% d'URL legítims, és essencial assegurar-se que aquest equilibri es manté quan es divideixen les dades en conjunts de formació i validació.

```python
X = data_preprocessada.drop('status', axis=1)
y = data_preprocessada['status']
```

Per a poder assegurar que els conjunts de dades d'entrenament i validació tenen aproximadament el mateix percentatge de mostres de cada classe objectiu que el conjunt de dades complet, aplicarem un mostreig estratificat

Utilitzarem un desglossament 80-20, on el 80% de les dades s'utilitza per a la formació i el 20% restant s'utilitza per a la validació. Això assegura que hi ha una quantitat suficient de dades per entrenar models robusts mentre encara hi ha un conjunt representatiu per a la validació.

```javascript
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=42)
```

4.2 Mètode d'avaluació del model

Validació creuada: Donada la mida del conjunt de dades (11.430 URLs), és factible utilitzar k-fold validació creuada per avaluar el rendiment del model. La validació creuada ens proporciona una millor estimació del rendiment d'un model en dades no vistes en comparació amb una divisió simple de train-test.

K-Fold Cross-Validation: Pel que fa a k-fold Cross-validation, el conjunt d'entrenament es divideix en "k" conjunts més petits. Per a cada un dels plecs "k", un model s'entrena utilitzant $k - 1$ dels plecs i validat en el plec restant. Aquest procés es repeteix "k" vegades, amb cada plec utilitzat exactament una vegada com a dades de validació. Per a obtenir una única puntuació, es realitza una mitjana amb els resultats donats.

En el nostre cas utilitzarem 10 plecs, ja que es un número recomanat i bastant comú. Per altra banda, com el conjunt de dades està bastant equilibrat, realitzarem una validació creuada k-fold estratificada per assegurar que cada plec sigui un bon representant del conjunt de dades general.

A més, al fer servir StratifiedKFold, ens assegurem que cada divisió del conjunt de dades manté la mateixa proporció de classes que el conjunt de dades original, cosa que és important per a conjunts de dades desequilibrats (no es el nostre cas) o quan la proporció de classes és crítica per a la predicció.

```python
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

4.3 Mètriques d'avaluació

Atès que el conjunt de dades és equilibrat, la precisió pot ser una mètrica fiable. No obstant això, en la detecció de phishing, tant els falsos positius (URLs legítims classificats com a phishing) com els falsos negatius (URLs de phishing classificats com a legítims) poden ser problemàtics. Per tant, considerarem altres mètriques.

Accuracy: És la proporció d'instàncies classificades correctament sobre el total d'instàncies.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + False\ Positives + True\ Negatives + False\ Negatives}$$

Com que el dataset està equilibrat (50% de URLs de phishing i 50% de URLs legítimes), l'accuracy pot ser una mètrica fiable. No obstant això, en detecció de phishing, és important no confiar únicament en l'accuracy, ja que un model podria simplement classificar tot com a "legítim" i encara aconseguir un 50% d'accuracy.

Recall (Sensibilitat): Mesura quantes de les instàncies positives reals han estat identificades correctament.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

El recall és crític en la detecció de phishing. Un valor baix de recall indicaria que moltes URLs de phishing s'estan classificant erròniament com a legítimes.

F1-Score: És la mitjana harmònica de precisión i recall, i proporciona un equilibri entre ambdues mètriques.

$$F1 = \frac{Precision \times Recall}{Precision + Recall}$$

Donat que tant la precisión com el recall són importants en la detecció de phishing, l'F1-score pot ser una mètrica útil per avaluar un model. Un F1-score elevat indica que el model té un bon equilibri entre precisión i recall.

5. Execució de diferents models de Machine Learning

En aquesta secció discutirem i analitzarem alguns dels diferents models de machine learning i els aplicarem al nostre dataset per a poder valorar quin obté els resultats més òptims

5.1 Naïve Bayes:

En primer lloc, Naïve Bayes, és un algorisme d'aprenentatge supervisat basat en l'aplicació del teorema de Bayes amb la suposició "naïve" d'independència entre les característiques. Aquest algorisme calcula la probabilitat d'una classe donada una observació (conjunt de característiques), i selecciona la classe amb la probabilitat més alta.

Algunes raons per les quals Naïve Bayes és adequat per al nostre cas d'ús són:

Velocitat: Es pot entrenar ràpidament, cosa que és útil donat que el nostre conjunt es d'unes característiques grans.

Escalabilitat: Maneja bé un gran nombre de característiques, que és comú en la detecció de phishing. En el nostre cas tenim 88 tipus diferents de caracterìstiques

Rendiment: Tot i la seva simplicitat, Naïve Bayes sovint funciona bé en la classificació binària, com és el cas amb la detecció de phishing (detectar si un url es phishing "1" o es legitim "0")

Per altra banda, per aplicar aquest algorisme all nostre dataset, no hem realitzat ajustos explícits dels paràmetres, degut a que la configuració predeterminada ha estat suficient per a aconseguir un rendiment òptim amb aquest conjunt de dades específic.

L'avaluació del model s'ha realitzat mitjançant Stratified Cross-Validation, la qual assegura que cada subconjunt de la validació manté la mateixa proporció de classes com el conjunt de dades original. Els resultats de l'avaluació han estat positius, seguidament analitzarem les mètriques obtingudes

Precisió (Accuracy): Hem obtingut una accuracy de 0,706% aixo indica que el model és capaç de classificar correctament aproximadament el 71% de les instàncies. Tot i que aquesta no és una mala puntuació, tampoc és excepcionalment alta, el que suggereix que hi ha espai per a millora.

Recall: El "recall" per a la classe 1 (phishing) és notablement baix (0.44), mentre que per a la classe 0 (legítima) és molt alt (0.97). Això implica que el model és molt bo detectant URLs legítimes, però no tan eficaç en detectar URLs de phishing.

Puntuació F1: La puntuació F1 per a la classe 1 és significativament més baixa que per a la classe 0. Aquesta mètrica combina precisió i "recall" i, per tant, una puntuació F1 més baixa per a la classe 1 suggerix un equilibri pitjor entre aquestes dues mètriques per a aquesta classe.

Pel que fa a la hipòtesi d'independència de variables a partir dels resultats obtinguts, podria ser no valida. El rendiment desigual entre les classes i la puntuació F1 relativament baixa per a la classe 1 suggereixen que podrien existir interdependències entre les característiques que afecten la capacitat del model de fer prediccions precises. Això destaca la necessitat d'analitzar més a fons les relacions entre les variables o considerar l'ús d'un model diferent que no assumeixi la independència de les variables. Es per aquest motiu que seguirem aplicant diferents models per a trobar millors resultats

5.2 K-NN:

Primerament, pel que fa a KNN, aquest es un és un algorisme supervisat que es pot utilitzar tant per a la classificació com per a la regressió. En la classificació, l'objectiu de K-NN és determinar a quina categoria pertany una nova observació basant-se en les observacions més propers a ella dins del conjunt de dades d'entrenament.

Algunes de les raons per les quals K-NN pot ser util per al nostre dataset son:

No Paramètric: Com que K-NN no assumeix res sobre la forma de la distribució de les dades, pot ser particularment útil en situacions on la relació entre les característiques és complexa i no es pot capturar fàcilment amb models paramètrics.

Eficaç amb Conjunts de Dades Petits: Tot i que el K-NN pot patir amb conjunts de dades molt grans degut al seu cost computacional, amb conjunts de dades de mida moderada pot ser molt efectiu, en el nostre cas es bastant efectiu degut al tamany "reduït" del dataset

Adaptatiu: El model es pot adaptar fàcilment a canvis en les dades. A mesura que es recull nova informació, el model K-NN pot incorporar aquesta informació sense la necessitat de ser reentrenat, cosa que és útil en un àmbit dinàmic com la detecció de phishing.

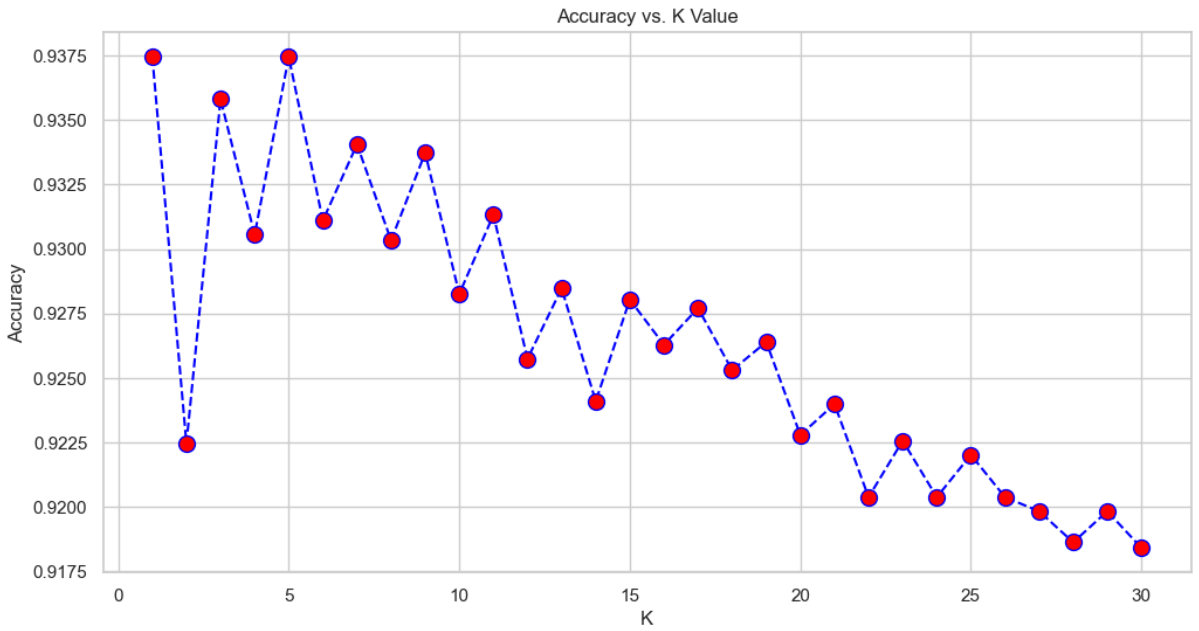
Útil per a Classificació Multiclasse: Tot i que en aquest cas el problema és de classificació binària, K-NN es pot aplicar sense cap modificació a problemes de classificació multiclasse, la qual cosa pot ser útil si el problema s'expandeix per incloure múltiples tipus de phishing o altres classes de problemes relacionats.

Interpretable: Les prediccions fetes per K-NN són fàcilment interpretables ja que simplement es basen en la majoria de vots dels veïns més propers. Això pot ser útil per a la transparència i la confiança en les prediccions dins d'organitzacions o aplicacions on les decisions necessiten ser explicades o justificades.

Sensible a Característiques Rellevants: Quan s'utilitza amb un conjunt de característiques ben seleccionades, K-NN pot ser molt potent perquè cada característica contribueix directament a la decisió de classificació basada en la similitud de les instàncies.

Per altra banda, pel que fa al **procediment de selecció de k**, en primer lloc, hem normalitzat les dades mitjançant ***StandardScaler***, aixo es crucial en K-NN, ja que aquest model és sensible a la magnitud de les característiques. Seguidament hem aplicat ***GridSearchCV*** amb una gamma de valors de *k* (de 1 a 30). Aquesta estratègia permet provar diferents valors de *k* i trobar el més òptim per al nostre

model. També, el paràmetre de validació creuada ***cv=skf*** assegura que cada conjunt de dades d'entrenament/test es faci servir de manera equitativa, aportant robustesa als resultats.



Com podem observar en l'imatge anterior, el millor valor de K ha estat ***k = 1***

Per altra banda, un cop obtingut el millor valor de K, procedim a entrenar el model K-NN

```python
knn_model = KNeighborsClassifier(n_neighbors=best_k)

knn_model.fit(X_train_scaled, y_train)

y_pred = knn_model.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

El model ha aconseguit una alta precisió (~93.48%), indicant un bon ajustament. A més, la matriu de confusió i l'informe de classificació mostren un bon equilibri entre les dues classes.

```
Precisió (Accuracy): 0.9348206474190727
Matriu de Confusió:
 [[1069   74]
 [  75 1068]]
Informe de Classificació:
              precision    recall  f1-score   support

         0.0       0.93      0.94      0.93      1143
         1.0       0.94      0.93      0.93      1143

    accuracy                           0.93      2286
   macro avg       0.93      0.93      0.93      2286
weighted avg       0.93      0.93      0.93      2286
```

Tot i que els valors obtinguts son bastant bons, explorarem diferents mesures de distancia per a obtenir millors resultats en el model

```python
k_range = range(1, 31)
distance_metrics = ['euclidean', 'manhattan', 'minkowski']

param_grid = {
  'knn__n_neighbors': k_range,
  'knn__metric': distance_metrics
}

pipeline = Pipeline([
  ('scaler', StandardScaler()),
  ('knn', KNeighborsClassifier())
])

grid_search = GridSearchCV(pipeline, param_grid, cv=skf, scoring='accuracy')

grid_search.fit(X_train, y_train)
```

Després d'ajustar k, em emprat diferents mesures de distància. A més de la mesura de distància euclidiana (per defecte), hem provat les mesures de Manhattan i

Minkowski. Aquesta exploració l'hem fet per veure si diferents formes de calcular la distància entre els punts podien millorar el rendiment del model. Seguidament, també hem utilitzat ***GridSearchCV*** amb validació creuada per avaluar diferents combinacions de k i la mesura de distància, buscant la combinació que maximitzés la precisió.

```
Millor combinació de paràmetres: {'knn__metric': 'manhattan', 'knn__n_neighbors': 7}
Millor precisió: 0.949911276918846
```

Pel que fa als resultats, en aquesta segona iteració del model hem obtingut una millora significativa en la precisió. La millor combinació ha estat ***k = 7*** i la mesura de distància de ***Manhattan***, obtenint una precisió de 0.9499.

5.3 Decision Trees:

Pel que fa al model Decision Tree, aquest funciona desglossant un conjunt de dades en parts més petites i més petites mentre desenvolupa un arbre de decisions associat. Cada node de l'arbre representa una característica, cada branca representa una regla de decisió, i cada fulla representa un resultat. El camí des de la base fins a cada fulla representa una seqüència de decisions que condueixen a una predicció final.

Fent referencia al nostre cas, el model Decision Tree, ens pot beneficiar per les següents caràcterístiques.

Facilitat d'Interpretació i Transparència: Els arbres de decisió són fàcils d'entendre i visualitzar, el que fa que les decisions del model siguin transparents. Això ens pot ajudar per arribar a veure quin és el motiu pel qual una pàgina s'ha classificat com a phishing.

Maneig de Dades No Lineals: Poden capturar relacions no lineals entre les característiques i la variable objectiu, cosa que ens pot ser útil en la detecció de phishing, on les relacions entre els atributs son complexes.

Identificació de Característiques Importants: ens ofereixen la capacitat d'identificar les característiques més importants per a la predicció, cosa que ens pot ajudar a entendre quins factors són més crítics en la determinació de si un lloc web és phishing.

Adaptabilitat a Dades Canviantes: Poden ser reentrenats fàcilment amb noves dades per adaptar-se a patrons emergents en llocs web de phishing, una característica important donada la naturalesa dinàmica dels atacs.

Prevenció de Sobreajustament: Amb la regulació adequada (com a limitar la profunditat de l'arbre), es pot prevenir el sobreajustament, assegurant que el model generalitza bé a noves dades.

Per altra banda, pel que fa a la ***selecció de paràmetres***, Per a l'ajustament del model d'Arbre de Decisió, hem utilitzat GridSearchCV amb validació creuada estratificada. Aquest mètode ens ha permès explorar diferents combinacions de paràmetres per trobar la configuració òptima. Els paràmetres ajustats són:

- ***max_depth***: La profunditat màxima de l'arbre. Hem provat amb valors [10, 20, 30, None], on "None" significa que no hi ha límit en la profunditat.

- ***min_samples_split***: El nombre mínim de mostres necessàries per dividir un node intern. Hem provat amb [2, 5, 10].

- ***min_samples_leaf***: El nombre mínim de mostres necessàries per ser una fulla (o node terminal) de l'arbre. Hem provat amb [1, 2, 4].

```python
dt = DecisionTreeClassifier(random_state=42)
param_grid = {
  'max_depth': [10, 20, 30, None],
  'min_samples_split': [2, 5, 10],
  'min_samples_leaf': [1, 2, 4]
}

grid_search_dt = GridSearchCV(dt, param_grid, cv=skf, scoring='accuracy')
grid_search_dt.fit(X_train, y_train)

best_params = grid_search_dt.best_params_
```

La combinació de paràmetres que va resultar en la major precisió va ser:

- max_depth = 10
- min_samples_leaf = 1
- min_samples_split = 5

```python
best_dt = DecisionTreeClassifier(**best_params, random_state=42)
best_dt.fit(X_train, y_train)

y_pred_dt = best_dt.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
```

Hem fet l'avaluació del model en base a l'accuracy en el conjunt de prova, obtenint un resultat de 0.9374. Això indica un alt grau de precisió en la predicció de les classes.

Tot seguit obtenim informació mes especifica sobre cadascúna de les regles que s'han executat a l'arbre

```python
feature_names = X.columns.tolist()
n_nodes = best_dt.tree_.node_count
children_left = best_dt.tree_.children_left
children_right = best_dt.tree_.children_right
feature = best_dt.tree_.feature
threshold = best_dt.tree_.threshold

def get_rules(node, depth):
  if (children_left[node] != children_right[node]):
    if feature[node] != -2:
      name = feature_names[feature[node]]
      threshold_value = threshold[node]
      print(f"{'|   ' * depth}If {name} <= {threshold_value:.2f}")
            get_rules(children_left[node], depth + 1)
            print(f"{'|   ' * depth}Else (if {name} > {threshold_value:.2f})")
            get_rules(children_right[node], depth + 1)
    else:
        print(f"{'|   ' * depth}Predicted class:
{best_dt.classes_[np.argmax(best_dt.tree_.value[node])]} with probability
{100.0 * np.max(best_dt.tree_.value[node]) /
np.sum(best_dt.tree_.value[node]):.2f}%")

print("Decision Tree Rules:")
get_rules(0, 0)


importances = best_dt.feature_importances_
indices = np.argsort(importances)[::-1]

print("Feature importances:")
for f in range(X_train.shape[1]):
    print(f"{f + 1}. feature {indices[f]} ({importances[indices[f]]})")
```

Amb el codi anterior obtenim una llista la cual ens mostra les importàncies de les característiques del model, amb la característica 79 (potser 'google_index') tenint la major importància seguida de la característica 80 (potser 'page_rank'). Això suggereix que aquestes dues característiques són les més predictives per a la classificació de les URLs com a legítimes o phishing.

Per altra banda, les regles de l'arbre de decisió mostren una sèrie de condicions que conduïxen a una classificació. Per exemple, una de les regles diu que si 'google_index' és menor o igual a 0.50, i 'page_rank' és menor o igual a 0.05, i altres

condicions específiques es compleixen, llavors la URL es classifica com a phishing amb una probabilitat del 100%. Això indica que URLs amb un baix índex de Google i un rang de pàgina baix tenen una alta probabilitat de ser phishing.

La descripció de les regles també ens mostra que hi ha fulles que prediuen amb una probabilitat del 100% per a una sola classe, el que podria indicar que aquestes fulles no tenen una barreja d'exemples positius i negatius, sinó que són "pures". Tanmateix, en altres fulles on hi ha una probabilitat menor que el 100%, és probable que hi hagi una barreja d'exemples positius i negatius, el que podria indicar que el model està menys segur en aquestes classificacions.

5.4 Support Vector Machines:

En primer lloc, les SVM són una potent tècnica d'aprenentatge supervisat utilitzada tant per a la classificació com per a la regressió. Són particularment conegudes per la seva eficàcia en classificar conjunts de dades complexos on les relacions entre les característiques no són lineals. Seguidament, esmentarem alguns dels aspectes clau del model, i com poden beneficiar al nostre conjunt de dades:

- ***Eficàcia en Espais de Gran Dimensió:*** Ja que el nostre conjunt de dades te moltes característiques, SVM pot ser particularment útil, ja que la seva eficàcia no disminueix significativament a mesura que augmenta el nombre de dimensions.

- ***Flexibilitat a través dels Nuclis:*** La capacitat d'utilitzar diferents nuclis fa que SVM sigui extremadament adaptable a diverses estructures de dades. Per a conjunts de dades no lineals, el nucli RBF pot ser particularment beneficiós (com es el nostre cas)

- ***Prevenció del Sobreajust:*** A través del paràmetre de regularització ***C***, SVM ens ofereix ofereix una bona balança entre l'aprenentatge del model i la prevenció del overfitting, que és crucial quan treballem amb conjunts de dades complexos.

Pel que fa a l'elecció del ***nucli i dels paràmetres***, hem emprat GridSearchCV, el qual ens ha retornat una llista amb els millors parametres.

```python
svm = SVC()

param_grid = {
  'kernel': ['linear', 'rbf', 'poly'],
  'C': [0.1, 1, 10],
  'gamma': ['scale', 'auto']
}

grid_search = GridSearchCV(svm, param_grid, cv=skf, scoring='accuracy')

grid_search.fit(X_train, y_train)
```

El nucli ***rbf (Radial Basis Function)***, ha estat seleccionat com el millor. Aquest nucli és molt popular per la seva capacitat de gestionar casos no lineals i per la seva flexibilitat en la formació de fronteres de decisió.

En quant als paràmetres ***C*** i ***gamma***. Hem obtingut els següents resultats:

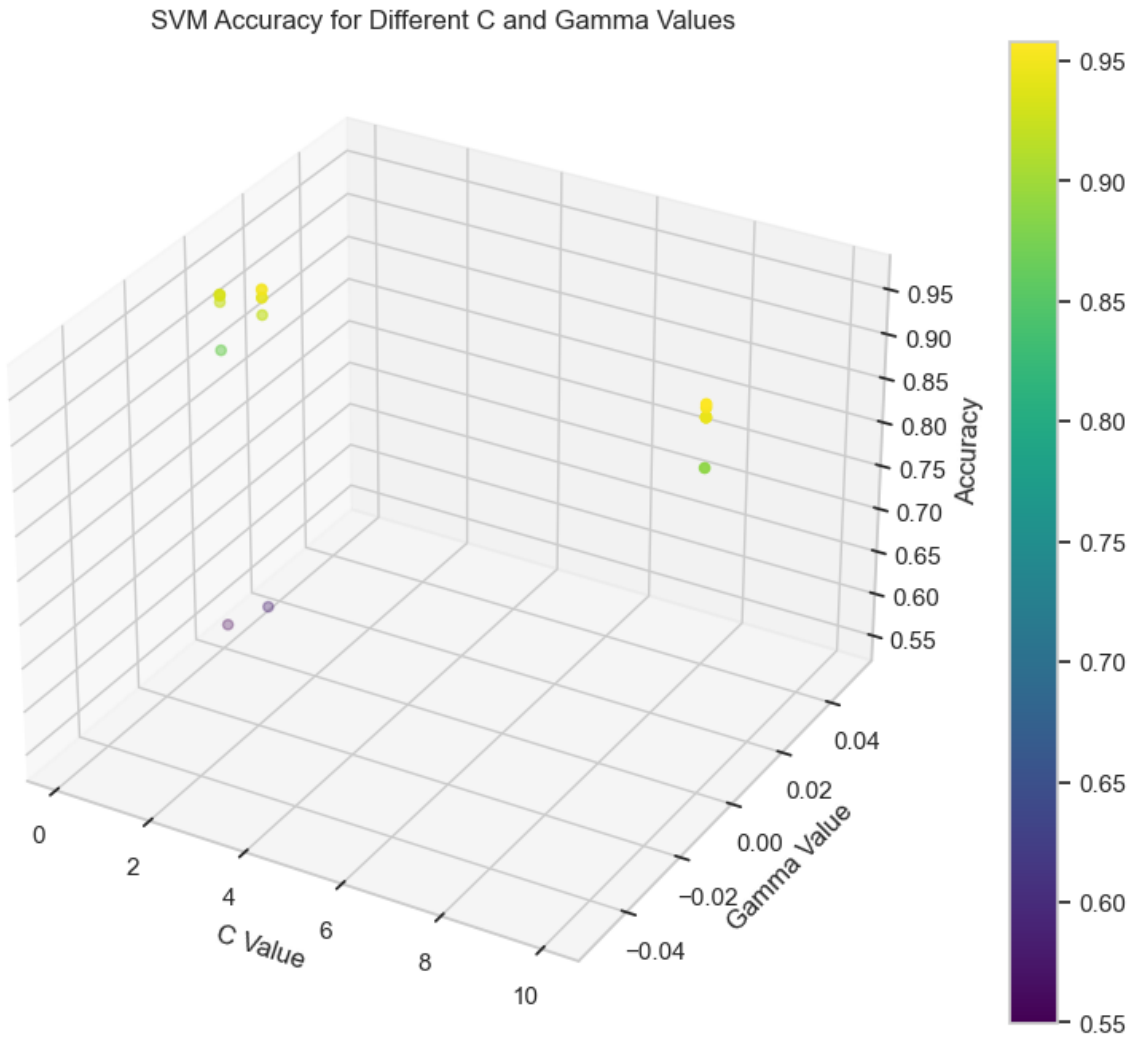
- ***C = 10:*** Aquest valor relativament alt indica una preferència per a un marge més petit amb una classificació més precisa dels exemples d'entrenament. Un C alt pot portar a un model més complex i, potencialment, a sobreajustar, però en aquest cas, sembla funcionar bé, donada l'alta precisió.

- ***gamma = 'scale'***: Aquesta opció adapta automàticament el paràmetre gamma a la distribució de les dades, oferint un bon equilibri entre complexitat i generalització.

```
Millors paràmetres: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Precisió (Accuracy): 0.9553805774278216
```

Cal destacar, que hem obtingut una accuracy de 0.9554, la qual es força alta (de moment la més alta pel que fa als mètodes anteriors)

Seguidament podem observar una representació gràfica dels paràmetres C, gamma i accuracy



Per altra banda, el nombre de vectors de suport és 1369. Aquests són els punts de dades crítics que formen la frontera de decisió. Aquest nombre relativament elevat indica que la frontera de decisió és complexa i ajustada a les nostres dades.

Inspeccionant els vectors de suport, podem entendre millor quins punts de dades són els més influents en la formació de la frontera de decisió. Per exemple, els valors mostrats en els vectors de suport indiquen quines característiques juguen un paper més important en la classificació.

```
Nombre de vectors de suport: 1369
Vectors de suport:
 [[0.053407   0.05714286 0.         ... 0.         1.         0.7       ]
 [0.00552486 0.03809524 0.         ... 0.         0.         0.2       ]
 [0.01411909 0.09047619 0.         ... 0.         0.         0.        ]
 ...
 [0.03806016 0.12380952 0.         ... 0.         1.         0.3       ]
 [0.0116636  0.08571429 0.         ... 0.         0.         0.5       ]
 [0.02946593 0.06666667 0.         ... 0.         0.         0.5       ]]
```

5.5 Meta-learning algorithms:

En el següent apartat, analitzarem diferents tipus de Meta-learning algorithms. Tot i això, en primer lloc, cal destacar que aquests, són tècniques que combinen múltiples models bàsics o "dèbils" per formar un model més potent i robust. Aquests mètodes son especialment beneficiaris en situacions complexes com en el nostre cas, la detecció de phishing, on la naturalesa dels dades es variada i amb subtileses difícils de capturar amb un únic model. Seguidament explorarem alguns dels meta-algoritmes més comuns.

5.5.1 Performance Majority Voting

En primer lloc, aquesta tecnica implica combinar les prediccions de diversos models de classificació i seleccionar la classe que la majoria dels models prediuen per a cada mostra. És una forma senzilla però eficaç de crear un Meta-learning algorithm. Per demostrar això, utilitzarem un exemple amb tres models diferents: un arbre de decisió, un classificador k-NN i un classificador SVM.

```python
model1 = DecisionTreeClassifier(max_depth=10, min_samples_split=5,
random_state=42)
model2 = KNeighborsClassifier(n_neighbors=1)
model3 = SVC(C=10, kernel='rbf', gamma='scale')
```

Pel que fa al primer model ***Decision Tree Classifier*** hem escollit els mateixos paràmetres que hem obtingut en l'analisi d'aquest model en concret ja que aixi podem tenir el millor rendiment amb uns parametres optims.

- ***max_depth=10***: Limita la profunditat de l'arbre. Una profunditat de 10 s'ha escollit per evitar sobreajustament i per permetre que l'arbre capturi suficient complexitat dels patrons de les dades.
- ***min_samples_split=5***: Especifica el nombre mínim de mostres requerides per dividir un node. Una valor de 5 ajuda a prevenir divisions massa fines que podrien portar a sobreajustament.
- ***random_state=42***: Assegura la reproductibilitat dels resultats mantenint la mateixa manera de generar l'arbre a cada execució.

Seguidament, el segon model escollit, ha estat ***K-NN*** i pel que fa als parametres, novament, hem escollit el mateix valor de K que hem trobat com a optim en l'apartat *5.2 K-NN.*

- ***n_neighbors=1***: S'utilitza el veí més proper per fer la predicció

Per últim, utilitzarem el model ***SVM*** emprant els parametres optims que hem trobat en l'apartat *5.4 Support Vector Machines*

- ***C=10***: Controla la penalització de les classificacions incorrectes. Un valor més alt de C significa que el model farà més esforços per classificar correctament totes les mostres d'entrenament (pot arribar a portar overfitting).
- ***kernel='rbf'***: El nucli RBF pot manejar casos on la relació entre les classes no és lineal.
- ***gamma='scale'***: Defineix quanta influència té una sola mostra d'entrenament, i 'scale' l'ajusta automàticament basat en el nombre de característiques.

Per últim, el model s'ha avaluat utilitzant cross-validation amb el StratifiedKFold, el que assegura una distribució uniforme de les classes en cada fold. Això és crucial per obtenir una avaluació precisa del model en diferents subconjunts de les dades.
L'accuracy mitjana obtinguda amb cross-validation ha estat de 0.958 (de moment la millor trobada). També cal destacar que tant el recall (0.955) com el F1-Score (0.958) son valors bastant grans per la qual cosa, molt positius per al nostre datset

```
Accuracy mitjana: 0.9587926509186351
Recall mitjana: 0.9559018039753593
F1 mitjana: 0.9586757266143173
```

5.5.2 Bagging

En primer lloc, cal destacar que el aquesta tècnica s'utilitza per a millorar l'estabilitat i la precisió de les màquines d'aprenentatge. Te el següent funcionament.

Primerament es comença amb la creació de múltiples conjunts de dades de formació. Aquests conjunts són generats mitjançant mostreig amb reemplaçament (bootstrap) del conjunt de dades original. Això significa que cada nou conjunt de dades pot tenir duplicats i alguns exemples del conjunt de dades original poden faltar.

Seguidament, cada un d'aquests conjunts de dades bootstrap s'utilitza per entrenar un model separat. Aquests models poden ser qualsevol classificador, però en el nostre cas utilitzarem arbres de decisió.

Per últim Un cop entrenats tots els models, les seves prediccions s'agreguen. En el cas de la classificació, això es fa normalment per Majority Voting.

En el noste cas pel que es refereix al estimador base, hem emprat un arbre de decisions, ja que aquest model ens permet capturar complexitats no lineals i interaccions entre caracteristiques. Hem seleccionat els parametres optims que hem trobat a l'apartat "*5.3 Decision Trees*" els quals son:

- *max depth = 10*
- *min_samples_split = 5*
- *random_state = 42*

```python
base_estimator = DecisionTreeClassifier(max_depth=10, min_samples_split=5, random_state=42)
```

Seguidament, hem aplicat el Bagging Classifier amb els seguents parametre:

- *n_estimators=100: Nombre d'arbres de decisió que formaran part de meta algoritme. Un nombre més gran d'estimadors pot millorar el rendiment, però també augmenta el temps de càlcul.*
- *random_state=42: Manté la consistència en la generació aleatòria per a la reproductibilitat.*

```python
bagging_model = BaggingClassifier(base_estimator=base_estimator, n_estimators=100, random_state=42)
```

Finalment, en cuant a l'avaluació del metode, hem emprat (com en la resta de casos), ***StratifiedKFold*** amb 10 splits i hem obtingut els següents resultats.

```python
cv_scores = cross_val_score(bagging_model, X, y, cv=skf, scoring='accuracy')
recall_scores = cross_val_score(bagging_model, X, y, cv=skf, scoring='recall')
f1_scores = cross_val_score(bagging_model, X, y, cv=skf, scoring='f1')
```

```
Accuracy mitjana: 0.9544181977252844
Recall mitjana: 0.9534536391804342
F1 mitjana: 0.9543724560275659
```

5.5.3 Random Forest

Pel que fa al model RandomForest aquest ens proporciona les següent avantatges:

- És robust davant overfitting, especialment quan es tracta de conjunts de dades amb un gran nombre de característiques. En el nostre cas, hem de tenir 87 característiques compte.

- Pot suportar bé les característiques categòriques i numeriques, tot i que en aquest cas ja hem transformat totes les característiques a numèriques.

- És fàcil de comprendre i interpretar, proporcionant la importància de les característiques que ens pot ajudar a entendre quins factors són més rellevants en la predicció de phishing.

- Funciona bé amb dades no balancejades, cosa que no és un problema en el nostre cos però és bo tenir en compte per altres contextos.

- És un model bastant flexible, que pot funcionar bé sense necessitat d'un ajust intensiu d'hiperparàmetres.

```python
rf_classifier = RandomForestClassifier(random_state=42)

accuracy= cross_val_score(rf_classifier, X_train, y_train, cv=skf,
scoring='accuracy')

recall = cross_val_score(rf_classifier, X_train, y_train, cv=skf,
scoring='recall')

f1_score = cross_val_score(rf_classifier, X_train, y_train, cv=skf,
scoring='f1')
```

Un cop executat el codi anterior, hem obtingut els següents resultats:

- Accuracy = 96.69%: Això significa que quan el model prediu que una URL és de phishing, és correcte aproximadament el 96.69% de les vegades. En termes de phishing, això indica que un percentatge molt alt d'URLs marcades com a malicioses realment són malicioses.

- Recall = 96.35%: Proporció d'URLs de phishing reals que el model ha estat capaç de detectar.

- F1-Score = 96.52%: Mitjana harmònica de la precision i el Recall, mesura robusta en el nostre cas per equilibrar la importància de la precision i el recall.

5.5.4 AdaBoost

En primer lloc, cal destacar que AdaBoost, és un mètode que busca construir un model fort a partir d'una sèrie de models més febles (generalment arbres de decisió d'un sol nivell, coneguts com a "stumps"). El procediment principal d'AdaBoost és ajustar els pesos de les instàncies de dades a cada iteració, donant més pes a les instàncies incorrectament classificades en iteracions anteriors. Això fa que el model resultant sigui adaptatiu, ja que s'enfoca en les instàncies més difícils de classificar.

Seguidament, en el nostre cas per a poder realitzar un anàlisi més profund i exhaustiu sobre AdaBoost, emprarem ***GridSearchCV*** per explorar una gamma més àmplia de paràmetres

```python
param_grid = {
  'n_estimators': [50, 100, 200],
  'learning_rate': [0.01, 0.1, 1],
  'base_estimator__max_depth': [1, 2, 3]
}

grid_search = GridSearchCV(ada_boost_model, param_grid, cv=skf,
scoring='accuracy')
grid_search.fit(X, y)

best_params = grid_search.best_params_
best_score = grid_search.best_score_
```

D'aquesta manera, obtenim que els millors paràmetres per aplicar al model son:

```
Millors paràmetres: {'base_estimator__max_depth': 3, 'learning_rate': 0.1, 'n_estimators': 200}
Millor puntuació (Accuracy): 0.9631671041119858
```

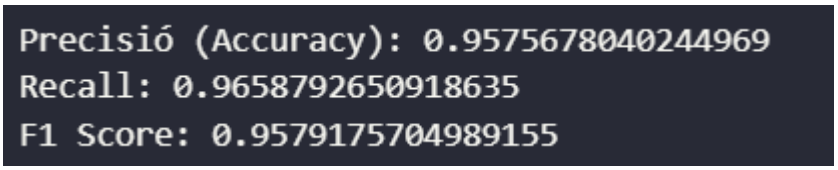
Per altra banda, un cop obtinguts els millors parametres, ja podem crear el model AdaBoost amb aquests

```python
best_ada_boost_model = AdaBoostClassifier(

base_estimator=DecisionTreeClassifier(max_depth=best_params['base_estimator_
_max_depth'], random_state=42),
  n_estimators=best_params['n_estimators'],
  learning_rate=best_params['learning_rate'],
  random_state=42
)
```

I seguidament, entrenem el model

```python
best_ada_boost_model.fit(X_train, y_train)

y_pred = best_ada_boost_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

Un cop entrenat el model, obtenim els següents resultats, els quals analitzarem en profunditat en l'ultim apartat



6. Comparacions i conclusions

En aquest apartat realitzarem una anàlisi exhaustiu i una comparació dels resultats obtinguts pels diferents mètodes de mineria de dades aplicats en el conjunt de dades de validació. La discussió se centrarà en tres mètriques clau: accuracy, recall i F1-Score. Las quals hem escollit en l'apartat ***"4. Criteri d'avaluació dels models de mineria de dades"***

Seguidament, mostrem una taula comparativa amb els resultats obtinguts per a cadascun dels mètodes:

Mètode	Accuracy	Recall	F1 Score
Naïve Bayes	70.69%	44.36%	60.19%
K-NN	94.99%	94.99%	94.99%
Decision Tree	93.74%	93.44%	93.48%
SVM	95.54%	95.63%	95.54%
Majority Voting	95.88%	95.59%	95.87%
Bagging	95.44%	95.35%	95.44%
Random Forest	96.77%	96.65%	96.71%
AdaBoost	95.76%	96.59%	95.79%

6.1 Anàlisi dels Resultats:

Naïve Bayes: Aquest model, basat en la suposició d'independència entre les característiques, ha mostrat limitacions en el context de la detecció de phishing. La seva baixa puntuació en recall (44.36%) indica una capacitat limitada per identificar correctament els casos de phishing, el que és crucial en aquesta aplicació. Aquestes limitacions podrien deure's a la interdependència i la complexitat de les característiques dins del conjunt de dades de phishing.

K-NN: Amb una precisió, recall i F1 Score al voltant del 95%, K-NN ha demostrat ser un model consistent i fiable. No obstant això, com que K-NN es basa en la proximitat de les característiques, la seva eficàcia pot dependre de la naturalesa del conjunt de dades. La variabilitat o el soroll en les dades podrien afectar negativament la seva capacitat de fer prediccions precises.

Decision Tree: Els arbres de decisió ofereixen una bona combinació de precisió (93.74%) i interpretabilitat. Són especialment útils perquè permeten visualitzar com es prenen les decisions, cosa que pot ser valuós per a l'avaluació de riscos en la detecció de phishing. No obstant això, poden estar més exposats al risc de overfitting en comparació amb altres models.

SVM: La força d'SVM en aquest context pot atribuir-se al seu ús de nuclis per a transformar l'espai de característiques, permetent una classificació més efectiva en casos no lineals. Aquesta capacitat pot ser particularment valuosa en dades de phishing, on les relacions entre característiques poden ser complexes.

Majority Voting (Meta-learning): Aquest enfocament combina les prediccions de diversos models per aconseguir un rendiment més equilibrat. El fet que superi la majoria dels models individuals subratlla la força d'aquest enfocament col·laboratiu, especialment en entorns amb diverses fonts de dades i models.

Bagging: Mostra un rendiment molt similar a Majority Voting. Aquesta tècnica, que utilitza el mostreig amb reemplaçament per crear diversos conjunts de dades i després combina els models, ajuda a reduir el overfitting, una consideració important en el modelatge de dades.

Random Forest: Aquest model ha obtingut la millor puntuació en totes les mètriques. Combina múltiples arbres de decisió per a millorar la precisió i reduir el risc de overfitting. Aquesta robustesa, juntament amb la seva capacitat per manejar una gran varietat de tipus de dades i la seva resistència al sobreajustament, el converteix en una opció ideal.

AdaBoost: Amb un alt recall (96.59%), AdaBoost demostra ser extremadament eficaç en la identificació de casos de phishing. Aquest model ajusta iterativament els pesos de les mostres per centrar-se en aquelles més difícils de classificar, millorant així la seva capacitat de detecció.

6.2 Consideracions Addicionals

La superioritat dels models de meta-aprenentatge en aquest estudi pot ser un indicador de la complexitat del problema de detecció de phishing, on diferents perspectives i enfocaments poden capturar millor les subtilitats de les dades.

La consistència entre les mètriques de validació i les obtingudes mitjançant cross-validation pot variar en funció de la complexitat del model i la mida del conjunt de dades.

6.3 Conclusió Personal

La tria de Random Forest com el millor mètode per aquesta tasca està basada no només en el seu rendiment superior en totes les mètriques clau, sinó també en la seva capacitat per oferir un model robust i generalitzable. La combinació de múltiples arbres de decisió permet al Random Forest capturar la complexitat i les interaccions entre les característiques de manera més eficient que els models individuals, fent-lo ideal per a la detecció de phishing. Aquesta capacitat d'adaptar-se a diverses formes de dades, juntament amb la seva resistència intrínseca al sobreajustament, el converteix en una eina poderosa i fiable per a la predicció en aquest domini. La seva aplicabilitat pràctica, juntament amb la seva alta precisió, recall i F1 Score, el fan la millor elecció per afrontar el repte de la detecció de phishing en aquest context específic.