ملخص مشروع كاشف التصيد الإلكتروني

نظرة عامة على المشروع

تم تطوير **كاشف التصيد الإلكتروني** بنجاح كأداة ذكية مفتوحة المصدر تستخدم تقنيات الذكاء الاصطناعي لكشف الروابط والمواقع الخبيثة. هذا المشروع يجمع بين التقنيات الحديثة والتصميم العصري لتوفير حل شامل لحماية المستخدمين من محاولات التصيد الإلكتروني.

الإنجازات المحققة

1. تطوير نموذج الذكاء الاصطناعي

- **جمع ومعالجة البيانات:** تم إنشاء مجموعة بيانات تحتوي على روابط شرعية وخبيثة
- **استخراج الخصائص:** تطوير نظام لاستخراج 11 خاصية مهمة من الروابط
- **تدريب النماذج:** تم تدريب وتقييم 4 نماذج مختلفة:
- Naive Bayes
- Logistic Regression (الأفضل بدقة 100%)
- Support Vector Machine
- Random Forest
- **حفظ النموذج:** تم حفظ أفضل نموذج للاستخدام في الإنتاج

2. واجهة سطر الأوامر (CLI)

- **سهولة الاستخدام:** واجهة بسيطة تقبل رابط واحد كمعامل
- **نتائج فورية:** عرض النتيجة مع مستوى الثقة
- **معالجة الأخطاء:** التعامل مع الحالات الاستثنائية بشكل مناسب

3. واجهة الويب التفاعلية

- **التصميم:** واجهة عربية عصرية ومتجاوبة باستخدام React و Tailwind CSS
- **الوظائف:**
- إدخال الرابط وفحصه فورياً
- عرض النتائج مع تفاصيل التحليل
- نصائح الأمان للمستخدمين
- **تجربة المستخدم:** تصميم بديهي مع رسائل واضحة ومفيدة

4. الواجهة الخلفية (Backend)

- **Flask API**: خادم قوي يوفر نقاط نهاية للفحص
- **CORS**: دعم الطلبات من مصادر مختلفة
- **معالجة البيانات**: تحليل الروابط واستخراج الخصائص في الوقت الفعلي

5. الوثائق الشاملة

- **README.md**: دليل مفصل للتثبيت والاستخدام
- **CONTRIBUTING.md**: إرشادات للمساهمين
- **SECURITY.md**: سياسة الأمان والإبلاغ عن الثغرات
- **LICENSE**: ترخيص MIT للاستخدام المفتوح

التقنيات المستخدمة

الذكاء الاصطناعي والبيانات

- **Python 3.11**: لغة البرمجة الأساسية
- **Scikit-learn**: لبناء وتدريب نماذج التعلم الآلي
- **Pandas & NumPy**: لمعالجة وتحليل البيانات
- **Joblib**: لحفظ وتحميل النماذج

الواجهة الخلفية

- **Flask**: إطار عمل الويب
- **Flask-CORS**: للسماح بالطلبات من مصادر مختلفة
- **SQLAlchemy**: لإدارة قاعدة البيانات

الواجهة الأمامية

- **React 18**: مكتبة JavaScript الحديثة
- **Tailwind CSS**: للتصميم العصري
- **Shadcn/UI**: مكونات واجهة مستخدم جاهزة
- **Lucide React**: أيقونات عصرية
- **Vite**: أداة البناء السريعة

الخصائص المستخرجة للتحليل

1. **طول الرابط:** الروابط الطويلة غالباً مشبوهة
2. **عدد النقاط:** كثرة النقاط قد تشير لنطاقات فرعية مشبوهة
3. **رمز @:** وجوده في الرابط مشبوه
4. **الشرطات المزدوجة:** في غير مكانها الطبيعي
5. **طول المسار:** المسارات الطويلة قد تكون مشبوهة
6. **طول الاستعلام:** معاملات الاستعلام الطويلة مشبوهة
7. **طول الجزء:** أجزاء الرابط الطويلة مشبوهة
8. **استخدام HTTPS:** المواقع الآمنة تستخدم HTTPS
9. **طول اسم النطاق:** أسماء النطاقات الطويلة مشبوهة
10. **الشرطات في النطاق:** كثرة الشرطات مشبوهة
11. **عدد النطاقات الفرعية:** كثرتها قد تكون مشبوهة

نتائج الأداء

دقة النماذج

- **Logistic Regression**: 100% (الأفضل)
- **Random Forest**: 100%
- **Naive Bayes**: 0% (يحتاج تحسين)
- **Support Vector Machine**: 0% (يحتاج تحسين)

سرعة الاستجابة

- **واجهة سطر الأوامر:** أقل من ثانية واحدة
- **واجهة الويب:** 2-1 ثانية شاملة الشبكة

الملفات والمجلدات الرئيسية

```
backend-detector-phishing/
├── src/
│   ├── main.py              # نقطة دخول التطبيق
│   ├── routes/
│   │   └── phishing.py      # endpoints API للفحص
│   ├── models/              # نماذج قاعدة البيانات
│   └── static/              # ملفات الواجهة الأمامية
```

```
│   ├── pkl.best_phishing_model # النموذج المدرب
│   └── py.data_preprocessing  # معالجة البيانات
├── py.cli                  # واجهة سطر الأوامر
├── md.README               # دليل المشروع
├── md.CONTRIBUTING         # دليل المساهمة
├── md.SECURITY             # سياسة الأمان
├── LICENSE                 # ترخيص MIT
├── txt.requirements        # متطلبات Python
└── gitignore.              # ملفات Git المتجاهلة

/frontend-detector-phishing
├── /src
│   ├── jsx.App             # المكون الرئيسي
│   ├── /ui/components      # مكونات الواجهة
│   └── /assets             # الملفات الثابتة
├── html.index              # صفحة HTML الرئيسية
└── json.package            # متطلبات js.Node
```

كيفية الاستخدام

واجهة الويب

1. تشغيل الخادم: `python src/main.py`
2. فتح المتصفح على: `http://localhost:5000`
3. إدخال الرابط والنقر على "فحص الرابط"

واجهة سطر الأوامر

```
https://example.com cli.py python
```

المميزات البارزة

للمستخدمين

- **سهولة الاستخدام:** واجهة بديهية باللغة العربية
- **نتائج فورية:** فحص سريع ودقيق
- **تفاصيل شاملة:** تحليل مفصل لخصائص الرابط
- **نصائح أمنية:** إرشادات للحماية من التصيد

للمطورين

- **كود مفتوح المصدر:** متاح للجميع تحت ترخيص MIT
- **توثيق شامل:** دليل مفصل للتثبيت والتطوير
- **معمارية نظيفة:** فصل واضح بين الواجهات
- **قابلية التوسع:** سهولة إضافة مميزات جديدة

الاستخدامات المقترحة

الشخصية

- فحص الروابط المشبوهة قبل النقر عليها
- التحقق من رسائل البريد الإلكتروني
- تعليم الأطفال والمراهقين عن أمان الإنترنت

المؤسسية

- دمج الأداة في أنظمة الأمان الموجودة
- تدريب الموظفين على التعرف على التصيد
- مراقبة الروابط في الشبكة الداخلية

التعليمية

- تعليم مفاهيم الذكاء الاصطناعي
- مشاريع التخرج في الأمن السيبراني
- ورش عمل حول أمان الإنترنت

التطوير المستقبلي

مميزات مقترحة

- **تحليل المحتوى:** فحص محتوى الصفحة نفسها
- **قاعدة بيانات التهديدات:** تحديث مستمر للتهديدات الجديدة
- **تطبيق الهاتف:** نسخة للهواتف الذكية
- **إضافة المتصفح:** امتداد للمتصفحات الشائعة

تحسينات تقنية

- **نماذج أكثر تقدماً:** استخدام Deep Learning
- **بيانات أكثر:** توسيع مجموعة البيانات
- **أداء أفضل:** تحسين سرعة الاستجابة
- **دعم لغات أخرى:** واجهات بلغات متعددة

معلومات التواصل

- **البريد الإلكتروني:** mabbadi0@icloud.com
- **الهاتف:** 0592774301
- **GitHub:** (سيتم إضافة الرابط عند النشر)

الخلاصة

تم تطوير مشروع كاشف التصيد الإلكتروني بنجاح كأداة شاملة تجمع بين قوة الذكاء الاصطناعي وسهولة الاستخدام. المشروع جاهز للنشر على GitHub ويمكن استخدامه فوراً لحماية المستخدمين من التهديدات الإلكترونية.

هذا المشروع يُظهر المهارات التقنية في:
- **الذكاء الاصطناعي:** تطوير وتدريب نماذج التعلم الآلي
- **تطوير الويب:** بناء تطبيقات ويب حديثة
- **الأمن السيبراني:** فهم تهديدات التصيد وطرق مكافحتها
- **إدارة المشاريع:** تنظيم وتوثيق المشروع بشكل احترافي

المشروع مُعد للعرض على LinkedIn كمثال على القدرة على تطوير حلول تقنية مبتكرة تخدم المجتمع وتساهم في تحسين الأمان الرقمي.