# Detekcija i klasifikacija malicioznih Android aplikacija

Emilija Opsenica
RA 108-2022

## Definicija problema:

Detekcija malicioznih Android aplikacija i njihova klasifikacija u jednu od 36 malicioznih familija, korišćenjem kombinacije statičkih i dinamičkih karakteristika.

## Motivacija:

Detekcija malicioznih aplikacija je veoma uspešna iz razloga što se većinom koriste poznate prodavnice aplikacija (npr. Google Play) i ove prodavnice vrše odličnu detekciju malicioznih aplikacija.

Međutim, zbog usvajanja Digital Markets Act-a (DMA), iOS mora dozvoliti korisnicima korišćenje alternativnih prodavnica koje često nemaju toliko dobro zaštitu [1].

Zbog ovoga se povećava rizik da korisnik nesvesno dođe u interakciju sa malicioznim aplikacijama. Detekcija ovih aplikacija često zahteva puno resursa, ali korišćenjem mašinske inteligencije, ovi resursi i vreme potrebno za ovakvu evaluaciju može biti značajno smanjeno.

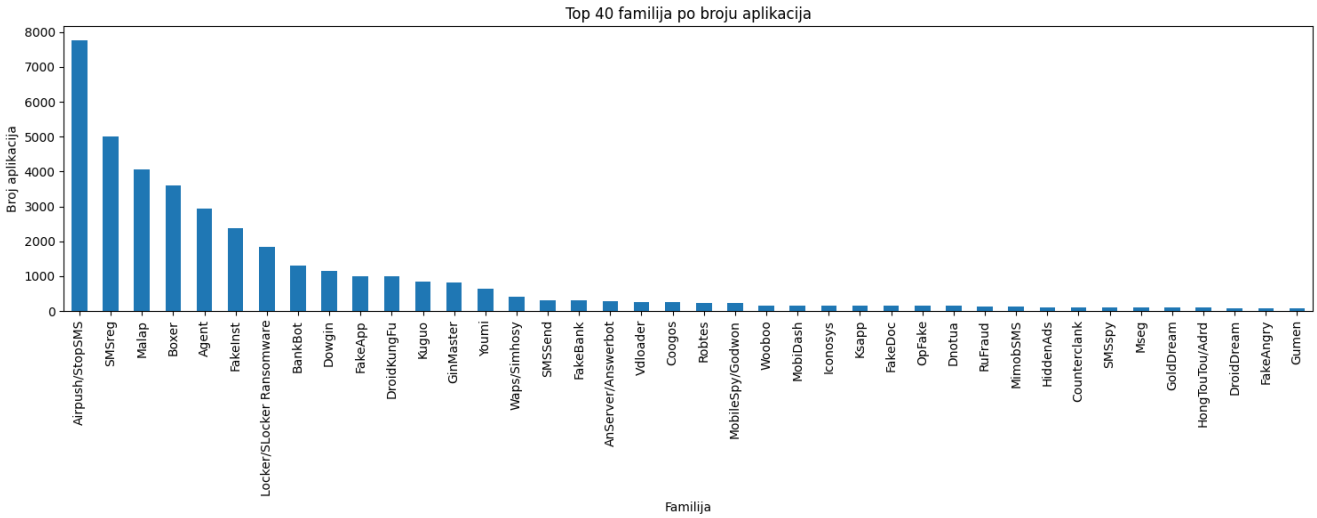
## Skup podataka:

Skup podataka koji ću koristiti je Kronodroid [2]. Ovaj repozitorijum kombinuje veliki broj javnih skupova podataka [3].

Detalji skupa podataka:

- Broj instanci: 41,382 malicioznih aplikacija (240 familija) i 36,755 benignih aplikacija. Ukupno: 78,137 redova.
- Atributi:
  - 289 dinamičkih atributa (npr. sistemski pozivi)
  - 200 statičkih atributa (npr. dozvole, filteri, meta podaci)
- Statički podaci obeležavaju nulom ili jedinicom da li se neka dozvola koristi dok dinamički podaci sadrže broj poziva funkcija ili API-a.
- Najznačajniji atributi su: *normal* (broj "normalnih" dozvola koje aplikacija zahteva), *dangerous* (broj "opasnih" permisija koje aplikacija zahteva).
- Ciljni atribut za detekciju je: *Malware* (0 ili 1)

- Ciljni atribut za klasifikaciju je: *MalFamily* (string)




## Način pretprocesiranja podataka:

Koristiću *LabelEncoder* iz *sklearn-a* da pretvorim nazive familija u numeričke oznake. Smanjiću broj obeležja koristeći PCA, izbaciti manje značajna obeležja kao naziv aplikacije i vreme i probaću da podelim podatke na statičke (koji se mnogo lakše dobijaju u realnom okruženju) i na dinamičke. Iz razloga što imam jako puno obeležja i podataka, koristiću Train/Val/Test raspodelu.

S obzirom da moj skup podataka ima nedostajućih vrednosti, sve redove koji nemaju vrednost za *MalFamily* ću obrisati, a ostale kolone koje imaju bar jednu nedostajuću vrednost (manje od 2% svih kolona) ću obrisati iz celog skupa podataka.

## Metodologija:

Napraviću dva modela koji se baziraju nad istim podacima. Prvi model će raditi detekciju da li je aplikacija maliciozna ili ne. Ulaz će biti par polja i atributa (većinski iz statičke analize), a izlaz će biti da li je aplikacija maliciozna ili ne.

Drugi model će koristiti podatke samo o malicioznim aplikacijama iz obe statičke i dinamičke analize fokusirajući se na dinamičku analizu.

Iz razloga što za veliki broj familija nema veliki broj podataka prisutnih, klasifikovaću aplikacije samo u familije koje imaju preko 100 aplikacija u skupu podataka.


Ovo će mi dati 35 familija koje pokrivaju 93% mojih podataka. Sve ostale će biti svrstane u „Other“ kategoriju.

Ideja je da ako nam prvi model kaže da je aplikacija maliciozna, da drugi model proba da je klasifikuje.

Probaću više modela koji se ponašaju dobro sa velikom količinom atributa: Random Forest, Support Vector Machine (SVM), Logistička regresija, Gradient Boosting (XGBoost, LightGBM), AdaBoost, MLP Classifier, Tabnet [4] i VAE.

### Način evaluacije:

Imaću Train/Val/Test raspodelu. Pokušaću koristiti raspodelu 70%/15%/15% za početak.
Za validaciju prvog modela koristiću preciznost, odziv i F1-score.

Za validaciju drugog modela koristiću F1-score i matricu konfuzije.

### Tehnologije:

Program ću pisati u *Python*-u. Potrebne biblioteke su: *pandas, matplotlib, seaborn, numpy, sklearn, xgboost, androguard* (za ekstrakciju podataka iz aplikacija – korisno za testiranje), *pathlib, concurrent.futures* i *multiprocessing* (za paralelizaciju i ubrzanje obrade velikog broja aplikacija).

### Relevantna literatura:

Primer urađenih projekata nad drugim skupom podataka:

1. https://www.kaggle.com/code/quackaddict7/detecting-android-malware-from-app-permissions
2. https://www.kaggle.com/code/gauranggupta123/android-malware-prediction-with-accuracy-99

Semi-nadgledana klasifikacija:

3. https://ieeexplore.ieee.org/document/9251198

Potencijalno proširenje projekta: RNN/LSTM za analizu sekvenci API poziva:

4. https://ijci.vsrp.co.uk/2025/03/malware-detection-for-android-systems-using-neural-networks/

### Reference:


[1] https://socradar.io/the-dangers-of-third-party-app-stores-risks-and-precautions/.

[2] https://github.com/aleguma/kronodroid.

[3] https://www.sciencedirect.com/science/article/pii/S0167404821002236.

[4] https://github.com/dreamquark-ai/tabnet.