Domaći zadatak 1

Izveštaj SMS poruke - da li su spam ili ne

1. Uvod

SMS Spam Collection je skup SMS označenih poruka koje su prikupljene za SMS Spam istraživanje. Sadrži jedan skup SMS poruka na engleskom jeziku od 5574 poruke, označene kao ham ili spam.

Kolekcija od 425 SMS spam poruka je ručno izvučena sa veb lokacije Grumbletekt. Ovo je forum u Velikoj Britaniji na kojem korisnici mobilnih telefona javno iznose tvrdnje o SMS spam porukama, većina njih bez prijavljivanja same spam poruke koju su primili. Identifikacija teksta spam poruka u tvrdnjama je veoma težak i dugotrajan zadatak, a podrazumevalo je pažljivo skeniranje stotina veb stranica.

Link do dataset-a: https://www.kaggle.com/datasets/marslinoedward/sms-spam-dataset

2. Učitavanje dataseta

Koristila sam “spam.cv” skup podataka. Encoding mora da se podesi zato što se radi o SMS porukama koje mogu da sadrže emoji i reči koje nisu na engleskom. Promenila sam nazive kolona skupa podataka koje će se koristiti, i izbacila sam kolonu koja ne mora da se uključi. Skup podataka nema nedostajućih vrednosti. Postojala su 403 duplirane vrednosti koje sam eliminisala. U ovom skupu podataka ima 4516 ham poruka i 653 spam poruka.

3. Nove kolone

Dodala sam nove kolone:

- num_characters: broj karaktera u porukama
- num_words: broj reči u porukama
- num_sentences: broj rečenica u porukama

Postoji ham poruka koja sadrži čak 910 karaktera. Koristila sam seaborn.pairplot funkciju da bih vizualizovala odnose između varijabli num_characters, num_words, num_sentences i ciljne varijable target. Može se primetiti:

- Većina poruka ima manji broj karaktera, manji broj reči i rečenica
- Poruke sa više karaktera obično imaju više reči i rečenica
- Nema jasne korelacije između broja reči i broja rečenica
- Spam poruke obično imaju više karaktera, reči i rečenica

Takođe sam dodala i kolonu koja predstavlja dužinu poruke. Može se videti da je dužina ham poruke obično manja od dužine spam poruke. Većina ham poruke ima dužinu ispod 100, za spam poruke je iznad 100.

4. Preprocesiranje

Za preprocesiranje teksta koristila sam funkciju “transform_text(text)” koja obuhvata sledeće korake:

- Konvertovanje teksta u mala slova - tekst se konvertuje u mala slova kako bi se obezbedilo da su sve reči iste (npr., "The" i "the" će biti tretirane kao iste reči)
- Tokenizaciju u pojedinačne reči - tekst se deli na pojedinačne reči (tokeni) koristeći NLTK-ovu funkciju “word_tokenize”
- Filtriranje numeričkih i alfanumeričkih tokena - petlja prolazi kroz sve tokenizovane reči i dodaje samo one koje su alfanumeričke (tj. koje sadrže samo slova i/ili brojeve)
- Uklanjanje stop reči i interpunkcija - **stopwords.words('english')** se koristi za dobijanje liste uobičajenih reči koje se često ne smatraju važnim za analizu (npr. "and", "the"). **string.punctuation** sadrži sve znakove interpunkcije. Petlja prolazi kroz listu “text” i dodaje u listu “y” samo one reči koje nisu u listi stop reči i koje nisu znakovi interpunkcije
- Stemming - Svaka preostala reč se skraćuje na njen osnovni oblik (stem) pomoću Porterovog stemmera (**ps.stem(i)**). Na primer, “running” se može svesti na “run”.
- Na kraju preostale reči su spojene u jedan string.

Koristila sam WordCloud kako bi vizualizovala najčešće reči u porukama. U spam porukama su to reči: “call”, “free”, “txt”, “text”,... U ham porukama su: “u”, “go”, “get”, “come”,...

5. Vektorizacija

Za ekstrakciju atributa koristila sam bag of words pristup. Potrebno je izbrojati pojavu svake reči. Svaki vektor će imati onoliko dimenzija koliko ima jedinstvenih reči u SMS korpusu. Prvo sam koristila SciKit Learn-ov CountVectorizer. Ovaj model će konvertovati kolekciju tekstualnih dokumenata u matricu broja tokena.

6. Klasifikacija

U alatu NLTK postoji nekoliko implementacija klasifikatora koje se mogu koristiti kao što su: NaiveBayes klasifikator, DecisionTree klasifikator, Maxent klasifikator.

Prvo sam podelila podatke na trening i test setove. Onda sam primenila funkciju za genirisanje atributa koja konvertuje vektor atributa u rečnik gde su ključevi nazivi atributa, a vrednosti su brojevi iz vektora. Ovo je korisno za pripremu podataka za NLTK klasifikatore koji često očekuju ulaz u obliku rečnika sa imenovanim atributima.

Rezultati:

- Naive Bayes Accuracy: 0.97807
- Decision Tree Accuracy: 0.95745
- Maxent Classifier Accuracy: 0.85816

NaiveBayes je u ovom slučaju najefikasniji, dok DecisionTree takođe daje veoma dobre rezultate. Maxent, iako često koristan, nije pokazao iste performanse kao prva dva modela u ovom konkretnom slučaju.