בחירת הטקסטים:

כשעברנו על הטקסטים בשני הקורפוסים, ראינו שיש הרבה טקסטים מתחום המשפט, ולכן חשבנו לבחור טקסטים מקבילים מהתנ"ך שהם קשורים למשפט גם כן. בחרנו בטקסטים מהתורה ובפרט מספרי שמות, ויקרא, במדבר, כי שם יש טקסטים בעלי אופי משפטים – חוקים. בדיעבד לאחר משימת תיוג הטקסטים לדעתנו הבחירה בטקסטים אלו הייתה שגויה, משום שאם היינו בוחרים טקסטים יותר מגוונים היינו יכולים לקבל תוצאות טובות יותר באופן כללי מכלל התנ"ך, שכן בטקסטים שבחרנו מהתנ"ך יש פעלים רבים ותבניות שחוזרות על עצמן.

עיבוד הטקסטים:

כדי להתחיל במשימת התיוג היה עלינו לשלוף מהקורפוסים שבחרנו את הפעלים הרלוונטיים לפי הבניין שבחרנו עשינו זאת בעזרת הסקריפטים tanchparser.py וwikiparser.py.

בניית מדריך תיוג:

בבניית מדריך התיוג נעזרנו בספרים של הבלשנים. ההסברים של גלינרט היה די ברור ומפורט עם דוגמאות ברורות, לעומת ההסברים של בלאו שהיו מעט פשטניים מדי לדעתנו. את מדריך התיוג ניתן לראות בקובץ tagging\_manual.docx.

תיוג הדוגמאות:  
תחילה התיוג היה מאוד מבלבל, אך עם ההתקדמות זה בא יותר בקלות. שיטת התיוג של גלינרט הייתה יותר קלה וברורה מאשר של בלאו, החלוקה הייתה יותר הגיונית. כמו כן, בשני השיטות יש תיוגים שברובם היו חופפים (לדוגמא תיוג 5 בשני השיטות, תיוג 1 ו2 בגלינרט היה חופף לתיוג 3 בבלאו, ותיוג 4 בשני המקרים). כמו כן, כפי שנראה בסטטיסטיקות על הדאטסט, פיזור התיוג היה לא מאוזן בכלל. דבר שיכול להשפיע על תוצאות המסווג – נראה זאת בהמשך.

בניית וקטור הדוגמאות:

החלטנו שכדי לקבל יותר מידע על הפעלים שברשותנו, אנחנו צריכים לדעת את הקונטקסט של כל פועל. לכן, כתבנו מתודות אשר בהינתן פועל, וגודל חלון, נותנות לנו את כל המילים שלפני ואחרי המילה כגודל החלון (לדוג', אם גודל החלון הוא 2, ניקח ' מילים קדימה ו 2 מילים אחורה). לכל פועל מכל הפעלים המתויגים, כתבנו את כל הנתונים הסינטקטיים והמורפולוגים לכל מילה אשר נמצאת בחלון. את נתוני הווקטור ניתן לראות בקובץ merged.xlsx

לאחר מכן החלטנו, שנשווה את הווקטורים שהוצאנו לשיפור נוסף שהוצע בהנחיות העבודה: הרכבי המאפיינים. הוספנו לכל וקטור הרכבים של מאפיינים עבור כל מילה בחלון. את ווקטורי המאפיינים האלה ניתן לראות בקוצים merged\_comb\_blau.xlsx וmerged\_comb\_glinert.xlsx.

לאחר מכן כאשר אימנו את המודל השתמשנו ב[OneHotEncoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html) מספריית scikit-learn לקודד את הדאטה, משום שהדאטה הוא ברובו קטגורי.

בהמשך נראה שהווקטור בעל ההרכבים של מאפיינים קיבל תוצאות טובות יותר במקצת.

את הפונקציונליות זו ניתן לראות בקבצים pds.py, feature\_vector.py

אימון המסווג:

תחילה הפרדנו את הדאטה למדגם אימון ומדגם טסט. לאחר מכן השתמשנו בספרייה [Lazy Predict](https://lazypredict.readthedocs.io/en/latest/usage.html#classification) כדי לבחור את המודל עם מדד הaccuracy הגבוה ביותר על מדגם הטסט. את התוצאות ניתן לראות בקובץ lazypredict\_results.txt.

עבור גלינרט ללא הרכב מאפיינים המודל הכי טוב שקיבלנו הוא [Random Forest](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html).

עבור בלאו ללא הרכב מאפיינים המודל הכי טוב שקיבלנו הוא [XGBoost](https://xgboost.readthedocs.io/en/stable/).

עבור גלינרט עם הרכב מאפיינים המודל הכי טוב שקיבלנו הוא [Extra Trees](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html).

עבור בלאו עם הרכב מאפיינים המודל הכי טוב שקיבלנו הוא [Extra Trees](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html).

לאחר מכן החלטנו לקחת את המודל שהניב את המדדים הכי טובים בכל גישה, ולעשות לו Hyperparameter tuning ו10 fold cross validation. זאת עשינו בעזרת ספריית [Optuna](https://optuna.org/), המשמשת לאופטימיזציה של היפר פרמטרים, תוך שימוש באופטימיזציה מבוססת SMBO כדי לחפש ביעילות את ערכי ההיפר פרמטרים הטובים ביותר. את תוצאות ההרצה ניתן לראות בקבצי הלוגים.