Elektromos autók népességi adatainak elemzése

Bevezetés

A Jupyter Notebookban található kóddal szerettem volna elemzést végzni az elektromos járműállomány adatain. A cél egy olyan osztályozási modell kialakítása, amely a járműgyártót jósolja elő különböző jellemzők alapján, mint például a modellév, a kezdő MSRP, az elektromos hatótávolság és a modell.

Adathalmaz

Az elemzéshez használt adathalmaz egy "Electric_Vehicle_Population_Data.csv" nevű CSV fájlban található. Az adathalmaz információkat tartalmaz az elektromos járművekről, beleértve a gyártót, a modellt, a modell évet, a kezdő MSRP-t és az elektromos hatótávolságot.

Ez a Jupyter Notebook részletes dokumentációját fogja nyújtani a küldött kódhoz. A dokumentáció magyarázatot ad minden lépésre, annak céljára és az azt megalapozó okokra.

Adatbetöltés és előfeldolgozás

1. Importálásra kerülnek a szükséges könyvtárak, beleértve a `pandast`, a `numpyt`, a `matplotlibet`, a `seaborn`-t és különböző modulokat a `scikit-learnből`. (Ezek megtalálhatóak a requirements fileban is, a könnyed telepítés érdekében)

2. Az adathalmaz betöltődik egy pandas DataFrame-be, amelyet `df`-nek nevezünk el, a `pd.read_csv()` függvény segítségével.

3. A többosztályos osztályozáshoz a célváltozó a 'Make' oszlop, amely a járműgyártót reprezentálja. Az `LabelEncoder`-t használjuk a célváltozó numerikus címkékre történő átalakításához, amelyet a `y` változóban tárolunk.

4. Meghatározzuk a kategóriába tartozó `'Model'` és a numerikus jellemzőket `'Model Year'`, `'Base MSRP'` és `'Electric Range'`.

5. Előfeldolgozási csővezetékek készülnek mind a numerikus, mind a kategorikus adatokhoz:

    - A numerikus adatokhoz létrehozunk egy `numerical_transformer` nevű csővezetéket, amely alkalmazza a MinMaxScaler-t az adatok [0, 1] tartományba skálázásához, majd a StandardScaler-t a standardizáláshoz.
    - A kategorikus adatokhoz az `OneHotEncoder`-t használjuk, `drop='first'` beállítással a dummy változó csapdát elkerülve, és `handle_unknown='ignore'` beállítással a ismeretlen kategóriák kezeléséhez a predikció során.

6. A numerikus és kategorikus adatok előfeldolgozási csővezetékei összecsomagolódnak a `ColumnTransformer` segítségével, és a `preprocessor` változóban tárolódnak.

Modellválasztás és képzés

1. A Support Vector Machine (SVM) modellt választottuk erre az osztályozási feladatra, és a `model` változóba tároljuk.

2. Az előfeldolgozási és modellezési lépések összecsomagolódnak egy `Pipeline`-ba, amelyet `pipeline`-nak nevezünk el.

3. Az adatot a scikit-learn `train_test_split` függvényével felosztjuk képzési és tesztelési halmazra, a teszteléshez pedig az adat 30%-át tartjuk fenn.

4. A hiperparaméterek finomhangolásához és a modell képzéséhez a GridSearchCV-t használjuk. A `param_grid` paramétert definiáljuk a különböző értékekkel az SVM hiperparamétereinek (`C`, `gamma`, és `kernel`) beállításához. A GridSearchCV kimerítő keresést végez a meghatározott paraméterkombinációk felett, 5-szörös keresztvalidációt alkalmazva.

5. A GridSearchCV által talált legjobb hiperparaméterek kiíratásra kerülnek.

Modell értékelése

1. A kiképzett modellt a teszthalmazon való előrejelzésekhez használjuk a `grid_search.predict()` függvény segítségével, és az pontossági pontszámot a scikit-learn `accuracy_score` függvényével számítjuk ki.

2. Egy egyedi `print_score` függvényt definiálunk, amely részletes értékelési metrikákat nyomtat ki mind a kiképzési, mind a tesztelési halmazokra. Tartalmazza a pontossági pontszámot, a besorolási jelentést és a zavaros mátrixot.

3. Az egyes SVM kernel típusok (lineáris, polinomiális és sugárirányfüggvény) teljesítményét az `print_score` függvény segítségével értékeljük.

Főkomponens-analízis (PCA)

1. A PCA alkalmazása adataink alacsonyabb dimenziós térben történő megjelenítésére. A scikit-learn `PCA` osztályát használjuk, `n_components=3` beállítással a dimenzionalitás 3 főkomponensre való csökkentéséhez.

2. Egy egyedi `SparseToDenseTransformer` transzformert definiálunk a ritka adatok sűrű formátumba való átalakításához, mivel a PCA sűrű bemenetet igényel.

3. A transzformált adatot egy szórási diagramon ábrázoljuk, ahol az első két főkomponens az tengelyek és a célváltozó a színezés alapja.

4. Újra használjuk a GridSearchCV-t a PCA-val együtt a csővezetékben. A `param_grid_pca` paramétert definiáljuk különböző hiperparaméter értékekkel az SVM-hez.

5. A GridSearchCV által talált legjobb hiperparaméterek kiíratásra kerülnek PCA-val.

6. Az SVM modell teljesítményét a PCA-val az `print_score` függvény segítségével értékeljük mind a kiképzési, mind a tesztelési halmazokra.

Következtetés

Ez a Jupyter Notebook bemutatja a többosztályos osztályozási modell létrehozásának folyamatát, amely a járműgyártó megjósolására szolgál különböző jellemzők alapján. Tartalmazza az adatok betöltését, előfeldolgozást, modell kiválasztást (SVM), hiperparaméterek hangolását a GridSearchCV segítségével,

valamint a modell kiértékelését. Emellett a dimenziócsökkentés és a vizualizáció érdekében alkalmazza a PCA-t. Kiértékelésre kerülnek a különböző SVM kernel típusok teljesítménye és a PCA hatása a modell teljesítményére.

A kód strukturált megközelítést nyújt az elektromos járműállomány adatainak osztályozási modellépítéséhez és kiértékeléséhez. Kiindulópontként szolgálhat további elemzésekhez és kísérletezésekhez különböző modellekkel, jellemzőkkel vagy hiperparaméterekkel.