Predikcija cena laptopa na osnovu karakteristika

PROJEKAT IZ PREDMETA RAČUNARSKA INTELIGENCIJA

Uvod

- **Tema:** Predviđanje cena laptopova korišćenjem mašinskog učenja
- **Cilj:** Razviti model(e) koji će predvideti cenu laptopova na osnovu specifikacija
- **Problematika:** Kupovina laptopa može biti izazovna zbog velikog broja dostupnih opcija sa različitim cenama.
- **Rešenje:** Primena tehnika mašinskog učenja za predviđanje cena na osnovu karakteristika laptopova
- **Okruženje:** Korišćenje Visual Studio Coda za razvoj i testiranje.

Arhitektura Rešenja

- **Podaci:** Korišćenje skupa podataka sa specifikacijama laptopova i njihovim cenama. (Kaggle.com)
- **Predprocesiranje Podataka:** Uklanjanje duplikata, konverzija tekstualnih atributa u numeričke, popunjavanje nedostajućih vrednosti.
- **Modeli Mašinskog Učenja:** Korišćenje više modela (Random Forest, Linearna Regresija, Decision Tree, Extra Trees, Ridge Regression) za predviđanje cena (odabir najboljeg modela) i njihova obuka na trening skupu podataka
- **Evaluacija performansi** svakog modela na osnovu metrika Mean Absolute Error (MAE) i R2-score (na test i validacionom skupu podataka).

Predprocesiranje podataka

- **Cilj:** Priprema podataka za mašinsko učenje.
- **Koraci:** Uklanjanje duplikata (metoda drop_duplicates iz pandas biblioteke), odvajanje ciljnog obeležja od ostalih atributa, konverzija tekstualnih atributa (npr. težina laptopa) u numeričke vrednosti, popunjavanje nedostajućih vrednosti, standardizacija podataka.
- Standardizacija je bitna u predprocesiranju podataka jer omogućava usklađivanje karakteristika (da se one u modelu tretiraju uravnoteženo), poboljšava performanse i stabilnost modela, i pomaže u pravilnoj interpretaciji rezultata analize. Bez standardizacije, mnogi algoritmi mogu loše funkcionisati ili dati nepovoljne rezultate.

Tehnički detalji

- **Pandas:** Učitavanje i manipulacija podacima.
- **Scikit-Learn:** Korišćenje `Pipeline`-a i `ColumnTransformer`-a za obradu i transformaciju podataka.
- SimpleImputer: Popunjava nedostajuće vrednosti sa najčešćim vrednostima.
- OneHotEncoder: Pretvara kategorijske vrednosti koje ne mogu biti direktno korišćene u modelima u binarne (jednobitne) vektore.
- StandardScaler: Standardizacija numeričkih karakteristika- transformisanje podataka tako da imaju srednju vrednost (mean) 0 i standardnu devijaciju (standard deviation) 1.

Modeli Mašinskog Učenja

- **Opis:** Upotreba više modela kako bi se postigla najbolja predikcija cena laptopova.
- **Modeli:**
- **Random Forest:** Kombinacija više stabala odlučivanja.
- **Linearna Regresija:** Jednostavan model zasnovan na linearnoj kombinaciji ulaznih karakteristika.
- **Decision Tree:** Drvo odlučivanja koje se deli na grane zasnovane na ulaznim karakteristikama.
- **Extra Trees:** Nasumična "šuma" stabala odlučivanja.
- **Ridge Regression:** Linearna regresija sa regularizacijo

Linearna regresija

- **Linearna regresija** je osnovni model regresije koji pokušava da uspostavi linearnu vezu između zavisne promenljive (Price_euros) i jedne ili više nezavisnih promenljivih (karakteristike laptopa). Model je definisan linearnom funkcijom oblika:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

```
from sklearn.linear_model import LinearRegression
# Inicijalizacija modela
models = {
    ...
    'Linear Regression': LinearRegression(),
    ...
}
# Treniranje modela
model.fit(X_train, y_train)
```

Implementacija u kodu

Decision Tree

- **Decision Tree** je model koji koristi strukturu drveta da donosi odluke na osnovu karakteristika podataka. Svaki čvor predstavlja odluku na osnovu karakteristika podataka, a grane predstavljaju ishode te odluke.
- Decision Tree može biti korišćen za **klasifikaciju** (dodeljivanje kategorija) ili **regresiju** (predviđanje kontinuiranih vrednosti).

Implementacija u kodu

```
from sklearn.tree import DecisionTreeRegressor
# Inicijalizacija modela
models = {
    ...
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    ...
}
# Treniranje modela
model.fit(X_train, y_train)
```

Random Forest

- **Random Forest** se zasniva na ansamblu (ensemble) metoda koje kombinuju predikcije više stabala odluka kako bi se poboljšala tačnost modela.
- Kombinuje predikcije više stabala tako što uzima prosek (za regresiju) ili većinsko glasanje (za klasifikaciju).

Implementacija u kodu

```
from sklearn.ensemble import RandomForestRegressor
# Inicijalizacija modela
models = {
    ...
    'Random Forest': RandomForestRegressor(random_state=42),
    ...
}
# Treniranje modela
model.fit(X_train, y_train)
```

Extra Trees

- **Extra Trees** je sličan Random Forest modelu, ali se razlikuje u načinu na koji bira tačke podele i podskupove podataka za svako stablo.
- Kombinovanje predikcija stabala korišćenjem nasumičnih podskupova karakteristika i nasumičnih tačaka podele, što vodi do smanjenja vremena treninga i potencijalno smanjenja prekomernog "overfittinga"*.

*Visoka preciznost na trening podacima, ali niska preciznost na test podacima

Implementacija u kodu

```
from sklearn.ensemble import ExtraTreesRegressor
# Inicijalizacija modela
models = {
    ...
    'Extra Trees': ExtraTreesRegressor(random_state=42),
    ...
}
# Treniranje modela
model.fit(X_train, y_train)
```

Ridge Regression

- **Ridge Regression** je vrsta linearne regresije koja koristi regularizaciju kako bi smanjila pretreniranost modela. Dodaje "kaznu" za veličinu koeficijenata u linearnom modelu, što može poboljšati performanse na test setu.

Matematički, cilj Ridge regresije je da minimizuje sledeću funkciju:

$$\text{Funkcija Cilja} = \text{Suma Kvadratnih Grešaka} + \alpha \times (\text{Suma Kvadrata Koeficijenata})$$

Što je preciznije:

$$\text{Funkcija Cilja} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{p} w_j^2$$

```
from sklearn.linear_model import Ridge
# Inicijalizacija modela
models = {
    ...
    'Ridge Regression': Ridge(),
    ...
}
# Treniranje modela
model.fit(X_train, y_train)
```

Implementacija u kodu

Validacija (Cross-Validation)

- **Cross-validacija**: Za svaki model:
- Model se trenira na trening setu.
- Funkcija cross_val_score se koristi za izvođenje 5-kratne cross-validacije (argument cv=5). To znači da će trening podaci biti podeljeni na 5 podskupova.
- U svakoj od 5 iteracija, model će biti treniran na 4 podskupa, a testiran na preostalom podskupu. Na taj način, svaki podskup će jednom biti korišćen za testiranje.
- Računaju se dva merila performansi: neg_mean_absolute_error (negativna srednja apsolutna greška) i r2 (R-kvadrat).
- Srednja vrednost ovih merila preko svih 5 iteracija se ispisuje kao rezultat cross-validacije za svaku metodu.

Evaluacija modela

- Svi ovi modeli su inicijalizovani i trenirani unutar `main()` funkcije. Nakon preprocesiranja podataka, podaci su podeljeni na trening i test setove. Svaki model se trenira na trening setu, a zatim se vrši evaluacija korišćenjem cross-validation (ukrštene validacije) i evaluacija na test setu.

Evaluacija modela

- Ovaj deo koda obezbeđuje trening, validaciju i evaluaciju performansi svakog modela na osnovu metrika Mean Absolute Error (MAE) i R2-score. Na kraju, za unos novih podataka od strane korisnika, koristi se najbolji model da bi se predvidela cena laptopa.

```
# Treniranje i evaluacija svakog modela
for model_name, model in models.items():
    print(f"Training {model_name}...")
    model.fit(X_train, y_train)

    # Cross-validation
    cv_scores_mae = cross_val_score(model, X_train, y_train, cv=5, scoring='neg_mean_absol
    cv_scores_r2 = cross_val_score(model, X_train, y_train, cv=5, scoring='r2')

    print(f'Cross-validation MAE: {-cv_scores_mae.mean()}')
    print(f'Cross-validation R2-score: {cv_scores_r2.mean()}')

    # Predikcija na test setu
    y_pred = model.predict(X_test)

    # Evaluacija na test setu
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'Test MAE: {mae}')
    print(f'Test R2-score: {r2}')
```

Mean Absolute Error (MAE)

- **Mean Absolute Error (MAE)** predstavlja prosečnu apsolutnu grešku između predviđenih vrednosti i stvarnih vrednosti. To je mera koliko su predikcije modela "daleko" od stvarnih vrednosti u proseku. **MAE** je jednostavna za interpretaciju jer je izražena u istim jedinicama kao i zavisna promenljiva. Niže vrednosti MAE ukazuju na bolju preciznost modela.
- Formula za MAE je:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Gde su:

- y_i stvarne vrednosti
- $\hat{y}_i$ predviđene vrednosti
- n ukupan broj uzoraka

Koeficijent determinacije (R2-Score)

- **R2-score** (koeficijent determinacije) predstavlja proporciju varijanse u zavisnoj promenljivoj koja je objašnjena nezavisnim promenljivama u modelu. R2-score pokazuje koliko dobro predikcije modela odgovaraju stvarnim vrednostima. Formula za R2-score je:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Gde su:

- y_i stvarne vrednosti
- $\hat{y}_i$ predviđene vrednosti
- $\bar{y}$ prosečna vrednost stvarnih vrednosti
- n ukupan broj uzoraka

Koeficijent determinacije (R2-Score)

- **R2-score** može da ima vrednosti od -∞ do 1:
- Vrednost 1 znači da model savršeno objašnjava varijansu u zavisnoj promenljivoj.
- Vrednost 0 znači da model ne objašnjava nikakvu varijansu u zavisnoj promenljivoj (predikcije su jednake proseku stvarnih vrednosti).
- Negativne vrednosti znače da model daje lošije predikcije od jednostavnog modela koji koristi prosečnu vrednost za predikciju.

Optimizacija hiperparametara

- Optimizacija hiperparametara je urađena pomoću **RandomizedSearchCV** i **GridSearchCV** tehnika

```
models_and_grids = {
    'Random Forest': {
        'model': RandomForestRegressor(random_state=42),
        'param_grid': {
            'n_estimators': [50, 100, 150],
            'max_depth': [None, 10, 20, 30],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4]
        },
        'search': RandomizedSearchCV,
        'search_params': {
            'n_iter': 10,
            'verbose': 1,
            'n_jobs': -1
```

Optimizacija hiperparametara

- Za svaki model, definisani su model, param_grid, search, i search_params:
- **model:** Instancira se model koji se optimizuje (npr. RandomForestRegressor, DecisionTreeRegressor, ExtraTreesRegressor, Ridge, Linear Regression).
- **param_grid:** Definiše mrežu hiperparametara koje treba istražiti. Ovo je lista ili raspon vrednosti za koje se model optimizuje.
- **search:** Specificira pretragu hiperparametara koju treba koristiti. U mom kodu, to je RandomizedSearchCV za četiri modela i GridSearchCV za Ridge regresiju.
- **search_params:** Parametri za pretragu hiperparametara, kao što su broj iteracija za RandomizedSearchCV i verbose nivo izlaza za obaveštavanje o napretku.

Optimizacija hiperparametara

- Optimizacija hiperparametara se vrši kroz pretragu mreže parametara:

```
model = config['model']
param_grid = config['param_grid']
search = config['search']
search_params = config['search_params']

print(f"\nTuning hyperparameters for {model_name}...")
if model_name == 'Ridge Regression':
    grid_search = search(estimator=model, param_grid=param_grid, cv=5, scoring='neg_me
else:
    grid_search = search(estimator=model, param_distributions=param_grid, cv=5, scorin
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_
best_models[model_name] = best_model
```

RandomizedSearchCV

- RandomizedSearchCV: Nasumično bira kombinacije hiperparametara iz zadatih mreža. Omogućava pretragu na temelju nasumičnih uzoraka što može biti brže i manje resursno zahtevno od Grid Search-a.

Parametri:

- n_iter: Broj nasumičnih kombinacija koje će se testirati.
- verbose: Kontrola nivoa obaveštenja o napretku pretrage.
- n_jobs: Broj paralelnih radnih niti za izvršavanje pretrage.

GridSearchCV

- GridSearchCV: Testira sve moguće kombinacije hiperparametara u datom param_grid. Precizniji, ali potencijalno skuplji u resursima.

Parametri:

- verbose: Kontrola nivoa obaveštenja o napretku pretrage.

GridSearchCV

- `GridSearchCV` koristim za pronalaženje najboljeg `alpha` parametra za `Ridge` regresiju

```
'Ridge Regression': {
    'model': Ridge(),
    'param_grid': {
        'alpha': [0.01, 0.1, 1, 10, 100, 1000, 10000]
    },
    'search': GridSearchCV,
    'search_params': {
        'verbose': 1
    }
}
```

Alpha parametar

- **Alpha** je parametar koji kontroliše **intenzitet regularizacije**. Regularizacija je tehnika koja dodaje dodatni član u funkciju cilja kako bi se smanjila kompleksnost modela i poboljšala njegova generalizacija na neviđene podatke.
- GridSearchCV pretražuje sve vrednosti alpha u param_grid koristeći kros-validaciju i Mean Absolute Error kao metriku performansi.
- U mom projektu se najbolja vrednost parametra alpha pokazala vrednost 1.

Analiza rezultata

- **Ridge Regression** je najbolji model pre i nakon optimizacije. Ima najmanju vrednost MAE i najvišu vrednost R2-score na test skupu. MAE je 169.07, a R2-score je 0.87, što ukazuje na visoku tačnost predikcija.
- **Random Forest** takođe pokazuje solidne performanse sa Test MAE od 186.45 i Test R2-score od 0.7478, što ga čini drugim najboljim modelom u ovom poređenju.

- Na osnovu dobijenih rezultata, najmanje uspešan model je **Decision Tree**.

(Test MAE: 218.90

Test R2-score: 0.6861)

Uspešnost optimizacije parametara

- **Random Forest:** Parametri su optimizovani, ali nije bilo značajnog poboljšanja u performansama. Test MAE je malo porasla (sa 186.45 na 190.62), dok je R2-score ostao praktično isti.
- **Decision Tree:** Optimzacija je dovela do pogoršanja performansi, što može značiti da je model previše prilagođen treniranju (overfitting). Test MAE se povećala, a R2-score značajno opao.
- **Extra Trees:** Optimizacija nije donela značajno poboljšanje. Test MAE je ostala gotovo ista, dok je R2-score malo opao.
- **Ridge Regression:** Parametri optimizacije su identični početnim vrednostima, što znači da Ridge Regression model već bio optimalno podešen pre hiperparametarske optimizacije.

Scikit-learn biblioteka

- Scikit-learn (sklearn) je popularna biblioteka za mašinsko učenje u Pythonu.

Scikit-learn uključuje širok spektar alata za:

- Predprocesiranje podataka (npr. imputer, scaler, encoder)
- Trening modela (regresija, klasifikacija, klasterizacija)
- Evaluaciju modela (metrike)
- Kros-validaciju
- Pipeline za automatizaciju procesa mašinskog učenja

Pandas

- Pandas je biblioteka za analizu podataka u Pythonu koja omogućava lako rukovanje i analizu strukturiranih podataka. Pandas pruža strukture podataka i funkcije dizajnirane za rad sa tabelarnim podacima, poput Excel tabela ili SQL baza podataka.
- Pandas biblioteka omogućava efikasnu manipulaciju i analizu podataka, što je ključni korak u pripremi podataka za modele mašinskog učenja.

Pandas biblioteka

- `SimpleImputer` za popunjavanje nedostajućih vrednosti.
- `StandardScaler` za standardizaciju numeričkih karakteristika.
- `OneHotEncoder` za enkodiranje kategorijalnih karakteristika.
- `ColumnTransformer` za primenu različitih predprocesora na različite skupove kolona.
- `Pipeline` za kombinovanje više koraka u jedan proces.
- `train_test_split` za deljenje podataka na trening i test skupove.
- `cross_val_score` za kros-validaciju modela

Predikcija nove cene laptopa

- **Unos sa konzole:** Omogućeno je korisnicima da unesu specifikacije laptopa koje nisu bile prisutne u skupu podataka.
- **Implementacija:** Korišćenje Python skripte sa `input` funkcijama za unos karakteristika laptopa.
- **Predviđanje Cena:** Primena prethodno obučenih modela za predviđanje cena na osnovu novih podataka.

Zaključak

- **Koristi:** Automatizacija i olakšavanje procesa predviđanja cena laptopova na osnovu specifikacija.
- **Napredak:** Implementacija više modela omogućava bolju procenu cene laptopova.
- **Budući Razvoj:** Mogućnost dodavanja novih modela, optimizacija performansi i proširenje funkcionalnosti.

HVALA NA PAŽNJI