Month 2: Data Analysis and Statistical Methods

Week 5: Exploratory Data Analysis (EDA)

Objective

Understand the importance of EDA and apply techniques to summarize datasets and derive meaningful insights using descriptive statistics and visualizations.

Tasks Completed

- Studied the basics of EDA, including summary statistics (mean, median, mode, std).
- Analyzed dataset using correlation analysis and visual tools.
- Client Project: Performed EDA on client's dataset and extracted key patterns.

Python Scripts

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('client_dataset.csv')
print(df.describe())

corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

sns.boxplot(data=df, x='category_column', y='value_column')
plt.title('Boxplot of Value by Category')
plt.show()
```

Key Learnings

- Understood how to clean and preprocess data.
- Applied descriptive statistics to interpret dataset characteristics.

- Visualized relationships using correlation plots and boxplots.

Conclusion

Exploratory Data Analysis gave a strong foundation in data understanding. The hands-on project helped generate actionable insights from real-world data.

Week 6: Probability and Statistical Testing

Objective

Study probability distributions and hypothesis testing methods to perform meaningful comparisons using statistical techniques.

Tasks Completed

- Learned about normal and Poisson distributions.
- Practiced t-test, chi-square test, and confidence intervals using SciPy.
- Client Project: Compared two business strategies using statistical testing.

Python Scripts

```
from scipy import stats
import numpy as np

data = np.random.normal(loc=50, scale=10, size=100)
print(f'Mean: {np.mean(data)}, Std: {np.std(data)}')

group1 = [22, 25, 27, 30, 35]
group2 = [28, 29, 33, 36, 40]
t_stat, p_val = stats.ttest_ind(group1, group2)
print(f'T-statistic: {t_stat}, P-value: {p_val}')
```

Key Learnings

- Learned to simulate distributions using NumPy.
- Understood hypothesis testing frameworks and interpreted p-values.
- Applied statistical analysis to real data for decision-making.

Conclusion

Statistical testing helped in drawing objective conclusions from data. The client project provided experience in comparing business metrics with statistical rigor.

Week 7: Introduction to Machine Learning

Objective

Learn the fundamentals of machine learning, types of ML algorithms, and build a basic model using scikit-learn.

Tasks Completed

- Studied supervised vs. unsupervised learning and types of ML models.
- Implemented a linear regression model using scikit-learn.
- Client Project: Built a price prediction model for the client dataset.

Python Scripts

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pandas as pd

data = pd.read_csv('house_prices.csv')
X = data[['area', 'bedrooms']]
y = data['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
```

Key Learnings

- Understood core ML concepts like regression, data splitting, and model fitting.
- Learned to evaluate model performance using MSE.

- Gained experience in applying ML to business-relevant data.

Conclusion

This week provided a solid start in ML. Building a predictive model enhanced understanding of algorithm application in real-world scenarios.

Week 8: Model Evaluation and Tuning

Objective

Understand evaluation metrics and improve model performance through cross-validation and hyperparameter tuning.

Tasks Completed

- Learned about accuracy, precision, recall, and F1-score.
- Used GridSearchCV for tuning model hyperparameters.
- Client Project: Tuned client prediction model to improve accuracy.

Python Scripts

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

X = data.drop('target', axis=1)
y = data['target']

model = RandomForestClassifier()
params = {'n_estimators': [50, 100], 'max_depth': [5, 10, None]}

grid = GridSearchCV(model, param_grid=params, cv=5)
grid.fit(X, y)

best_model = grid.best_estimator_
y_pred = best_model.predict(X)
print(classification_report(y, y_pred))
```

Key Learnings

- Learned about precision-recall tradeoff and evaluation metrics.

- Used GridSearchCV to systematically find the best parameters.
- Improved model performance significantly through tuning.

Conclusion

Model tuning allowed for better performance and robustness. The client project helped apply theoretical knowledge into impactful results.